Alex Dialogue Systems Framework Documentation

Release 0.1

2012-2013, UFAL-DSG, MFF, CUNI, CZ and other contributors

September 16, 2015

Contents

1 Contents **3**
1.1 Index of manually written in source code tree documentation 3

2 Alex modules **31**
2.1 alex package 31

3 Indices and tables **115**

4 How to write documentation **117**

Bibliography **119**

Python Module Index **121**

Alex Dialogue Systems Framework or simply Alex is a set of algorithms, classes, and tools to facilitate building spoken dialogue systems.

CHAPTER 1

Contents

1.1 Index of manually written in source code tree documentation

1.1.1 Building the language model for the Public Transport Info telephone service (Czech)

```
*WARNING* To build the language model, you will need a machine with a lot of memory (more than 16GB R
```

The data

To build the domain specific language model, we use the approach described in *Approach to bootstraping the domain specific language models*. So far, we have collected this data:

1. selected out-of-domain data - more than 2000 sentences
2. bootstrap text - 289 sentences
3. indomain data - more than 9000 sentences (out of which about 900 of the sentences are used as development data)

Building the models

The models are built using the `build.py` script.

It requires to set the following variables:

```
bootstrap_text                  = "bootstrap.txt"
classes                         = "../data/database_SRILM_classes.txt"
indomain_data_dir               = "indomain_data"
```

The variables description:

- bootstrap_text - the bootstrap.txt file contains handcrafted in-domain sentences.
- classes - the `../data/database_SRILM_classes.txt` file is created by the `database.py` script in the `alex/applications/PublicTransportInfoCS/data` directory.
- indomain_data_dir - should include links to directories containing `asr_transcribed.xml` files with transcribed audio data

The process of building/re-building the LM is:

```
cd ../data
./database.py dump
cd ../lm
./build.py
```

Distributions of the models

The `final.*` models are large. Therefore, they should be distributed online on-demand using the `online_update` function. Please do not forget to place the models generated by the `./build.py` script on the distribution servers.

Reuse of build.py

The `build.py` script can be easily generalised to a different language or different text data, e.g. the in-domain data.

1.1.2 Description of resource files for ASR

This directory contains acoustic models for different languages and recording conditions. It is assumed that only one acoustic model per language will be build.

However, one can build different acoustic models for different recording settings, e.g. one for VOIP and the other for desktop mic recordings.

Up to now, only VOIP acoustic models have been trained.

1.1.3 Description of resource files for VAD

Please note that to simplify deployment of SDSs, the the VAD is trained to be language independent. That means that VAD classifies silence (noise, etc.) vs. all sounds in any language.

At this moment, the `alex/resources/vad/` has only VAD models build using VOIP audio signal. The created models include:

- GMM models
- NN models

More information about the process of creating the VAD models is available in *Building a voice activity detector (VAD)*.

Please note that the NN VAD is much better compared to GMM VAD. Also `alex/resources/vad/` stores the models, but they should not be checked in the repository anymore. Instead, they should be on the online_update server and downloaded from it when they are updated. More on online update is available in *Online distribution of resource files such as ASR, SLU, NLG models*.

1.1.4 Public Transport Info, Czech - telephone service

Running the system at UFAL with the full UFAL access

There are multiple configuration that can used to run the system. In general, it depends on what components you want go use and on what telephone extension you want to run the system.

Within UFAL, we run the system using the following commands:

- `vhub_live` - deployment of our live system on our toll-free phone number, with the default configuration

- vhub_live_b1 - a system deployed to backup the system above
- vhub_live_b2 - a system deployed to backup the system above
- vhub_live_kaldi - a version of our live system explicitly using Kaldi ASR

To test the system we use:

- vhub_test - default test version of our system deployed on our test extension, logging locally into ../call_logs
- vhub_test_google_only - test version of our system on our test extension, using Google ASR, TTS, Directions, logging locally into ../call_logs
- vhub_test_google_kaldi - test version of our system on our test extension, using Google TTS, Directions, and Kaldi ASR, logging locally into ../call_logs
- vhub_test_hdc_slu - default test version of our system deployed on our test extension, using HDC SLU, logging locally into ../call_logs
- vhub_test_kaldi - default test version of our system deployed on our test extension, using KALDI ASR, logging locally into ../call_logs
- vhub_test_kaldi_nfs - default test version of our system deployed on our test extension, using KALDI ASR and logging to NFS

Running the system without the full UFAL access

Users outside UFAL can run the system using the following commands:

- vhub_private_ext_google_only - default version of our system deployed on private extension specified in private_ext.cfg, using Google ASR, TTS, Directions, and KALDI ASR, logging locally into ../call_logs
- vhub_private_ext_google_kaldi - default version of our system deployed on private extension specified in private_ext.cfg, using Google TTS, Directions, and KALDI ASR, logging locally into ../call_logs

If you want to test the system on your private extension, then modify the private_ext.cfg config. You must set your SIP domain including the port, user login, and password (You can obtain a free extension at http://www.sipgate.co.uk). Please make sure that you do not commit your login information into the repository.

```
config = {
        'VoipIO': {
                # default testing extesion
                'domain':   "*:5066",
                'user':     "*",
                'password': "*",
        },
}
```

Also, you will have to create a "private" directory where you can store your private configurations. As the private default configuration is not part of the Git repository, please make your own empty version of the private default configuration as follows.

```
mkdir alex/resources/private
echo "config = {}" > alex/resources/private/default.cfg
```

1.1.5 UFAL Dialogue act scheme

The purpose of this document is to describe the structure and function of dialogue acts used in spoken dialogue systems developed at UFAL, MFF, UK, Czech Republic.

Definition of dialogue acts

In a spoken dialogue system, the observations and the system actions are represented by dialogue acts. Dialogue acts represent basic intents (such as inform, request, etc.) and the semantic content in the input utterance (e.g. type=hotel, area=east). In some cases, the value can be omitted, for example, where the intention is to query the value of a slot e.g. `request(food)`.

In the UFAL Dialogue Act Scheme (UDAS), a dialogue act (DA) is composed of one or more dialogue act items (DAI). A dialogue act item is defined as a tuple composed of a dialogue act type, a slot name, and the slot value. Slot names and slot values are domain dependent, therefore they can be many. In the examples which follows, the names of the slots and their values are drown form a information seeking application about restaurants, bars and hotels. For example in a tourist information domain, the slots can include “food” or “pricerange” and the values can be such as “Italian”, “Indian” or ”cheap”, “midpriced”, or “expensive”.

This can be described in more formal way as follows:

```
DA = (DAI)+
DAI = (DAT, SN, SV)
DAT = (ack, affirm, apology, bye, canthearyou, confirm,
    iconfirm, deny, hangup, hello, help, inform, negate,
    notunderstood, null, repeat, reqalts, reqmore, request,
    restart, select, thankyou)
```

where SN denotes a slot name and SV denotes a slot value.

The idea of dialogue comes from the information state update (ISU) approach of defining a dialogue state. In ISU, a dialogue act is understood as a set of deterministic operations on a dialogue state which which result in a new updated state. In the UFAL dialogue act scheme, the update is performed on the slot level.

The following explains each dialogue act type:

```
ack         - "Ok" - back channel
affirm      - simple "Yes"
apology     - apology for misunderstanding
bye         - end of a dialogue - simple "Goodbye"
confirm     - user tries to confirm some information
canthearyou - system or user does not hear the other party
deny        - user denies some information
hangup      - the user hangs up
hello       - start of a dialogue - simple "Hi"
help        - request for help
inform      - user provides some information or constraint
negate      - simple "No"
null        - silence, empty sentence, something that is not possible to
                interpret, does nothing
```

`null` It can be also used when converting a dialogue act item confusion network into an N-best list to hold all the probability mass connected with all dialogue acts which were not added to the N-best list. In other words probability mass of pruned DA hypotheses.

```
notunderstood - informs that the last input was not understood
repeat      - request to repeat the last utterance
irepeat     - repeats the last utterance
```

```
reqalts     - ask for alternatives
reqmore     - ask for more details
request     - user requests some information
restart     - request a restart of the dialogue
select      - user or the system wants the other party to select between
                two values for one slot
thankyou    - simple "Thank you"
```

NOTE: Having this set of acts we cannot confirm that something is not equal to something, e.g. `confirm(x!=y)` → `confirm(pricerange != 'cheap')` → "Isn't it cheap?" If we used `confirm(pricerange = 'cheap')` then it means "Is it cheap?" In both cases, it is appropriate to react in the same way e.g. `inform(pricerange='cheap')` or `deny(pricerange = 'cheap')`.

NOTE: Please note that all slot values are always placed in quotes `"`.

Dialogue act examples

This section presents examples of dialogue acts:

```
ack()                           'ok give me that one'
                                'ok great'

affirm()                        'correct'
                                'erm yeah'

appology()                      'sorry'
                                'sorry I did not get that'

bye()                           'allright bye'
                                'allright then bye'

canthearyou()                   'hallo'
                                'are you still there'

confirm(addr='main square')     'erm is that near the central the main square'
                                'is it on main square'

iconfirm(addr='main square')    'Ack, on main square,'


iconfirm(near='cinema')         'You want something near cinema'

deny(name='youth hostel')       'not the youth hostel'

deny(near='cinema')             'ok it doesn't have to be near the cinema'

hello()                         'hello'
                                'hi'
                                'hiya please'

help()                          'can you help me'


inform(='main square')          'main square'

inform(addr='dontcare')         'i don't mind the address'
```

```
inform(food='chinese')          'chinese'
                                'chinese food'
                                'do you have chinese food'

negate()                        'erm erm no i didn't say anything'
                                'neither'
                                'no'

null()                          '' (empty sentence)
                                'abraka dabra' (something not interpretable)

repeat()                        'can you repeat'
                                'could you repeat that'
                                'could you repeat that please'

reqalts()                       'and anything else'
                                'are there any other options'
                                'are there any others'

reqmore()                       'can you give me more dtails'

request(food)                   'do you know what food it serves'
                                'what food does it serve'

request(music)                  'and what sort of music would it play'
                                'and what type of music do they play in these bars'

restart()                       'can we start again please'
                                'could we start again'

select(food="Chinese")&select(food="Italian)
                                'do you want Chinese or Italian food'

thankyou()                      'allright thank you then i'll have to look somewhere else'
                                'erm great thank you'
```

If the system wants to inform that no venue is matching provided constraints, e.g. "There is no Chinese restaurant in a cheap price range in the city centre" the system uses the `inform(name='none')` dialogue acts as in

Utterance: There is no Chinese restaurant in a cheap price range in the city centre"

Dialogue act: `inform(name='none')&inform(venue_type='restaurant')&inform(food_type='Chinese')&i`

There are examples of dialogue acts composed of several DAIs:

```
reqalts()&thankyou()            'no thank you somewhere else please'

request(price)&thankyou()       'thank you and how much does it cost'
                                'thank you could you tell me the cost'

affirm()&inform(area='south')&inform(music='jazz')&inform(type='bar')&request(name)
                                'yes i'd like to know the name of the bar in the south part of town t
                                'yes please can you give me the name of the bar in the south part of

confirm(area='central')&inform(name='cinema')
                                'is the cinema near the centre of town'


deny(music='pop')&inform(music='folk')
```

```
                                    'erm i don't want pop music i want folk folk music'


hello()&inform(area='east')&inform(drinks='cocktails')&inform(near='park')&inform(pricerange='dontca
                                    'hi i'd like a hotel in the east of town by the park the price doesn
```

An example dialogue form tourist information domain is in the following table:

Turn	Transcription	Dialogue act
System	Hello. How may I help you?	`hello()`
User	Hi, I am looking for a restaurant.	`inform(venue="restaurant")`
System	What type of food would you like?	`request(food)`
User	I want Italian.	`inform(food="Italian")`
System	Did you say Italian?	`confirm(food="Italian")`
User	Yes	`affirm()`

Semantic Decoding and Ambiguity

Very often there are many ways as to map (to interpret) a natural utterance into a dialogue act, , some times because of natural ambiguity of a sentence – sometimes because of the speech recognition errors. Therefore, a semantic parser will generate multiple hypotheses. In this case, each hypothesis will be assigned a probability meaning the likelihood of being correct and the dialogue manager will resolve this ambiguity in the context of the dialogue (e.g. other sentences).

For example, the utterance "I wan an Italian restaurant erm no Indian" can be interpreted as:

```
inform(venue="restaurant")&inform(food="Italian")&deny(food=Indian)
```

or:

```
inform(venue="restaurant")&inform(food="Indian")
```

In the first case, the utterance is interpreted that the user wants Italian restaurant and does not want Indian. However, in the second case, the user corrected what he just mistakenly said (that he wants Indian restaurant).

Please remember that semantic parsers should interpret an utterance only on the information present in the sentence. It is up to the dialogue manager to interpret it in the context of the whole dialogue:

```
inform(type=restaurant)&inform(food='Chinese')
'I want a Chinese restaurant'

inform(food='Chinese')
'I would like some Chinese food'
```

In the first case, the user explicitly says that he/she is looking for a restaurant. However, in the second case, the user said that he/she is looking for some venue serving Indian food which can be both a restaurant or only a take-away.

Building a statistical SLU parser for a new domain

From experience, it appears that the easiest approach to build a statistical parser for a new domain is to start with build a handcrafted (rule based) parser. There are several practical reasons for that:

1. a handcrafted parser can serve as a prototype module for a dialogue system when no data is available,
2. a handcrafted parser can serve as a baseline for testing data driven parsers,
3. a handcrafted parser in information seeking applications, if well implemented, achieves about 95% accuracy on transcribed speech, which is close to accuracy of what the human annotators achieve,

4. a handcrafted parser can be used to obtain automatic SLU annotation which can be later hand corrected by humans.

To build a data driven SLU, the following approach is recommended:

1. after some data is collected, e.g. a prototype of dialogue system using a handcrafted parser, the audio from the collected calls is manually transcribed and then parsed using the handcrafted parser,
2. the advantage of using automatic SLU annotations is that they are easy to obtain and reasonably accurate only several percent lower to what one can get from human annotators.
3. if better accuracy is needed then it is better to fix the automatic semantic annotation by humans,
4. then a data driven parser is trained using this annotation

Note that the main benefit of data driven SLU methods comes from the ability to robustly handle erroneous input. Therefore, the data driven SLU should be trained to map **the recognised speech** to the dialogue acts (e.g. obtained by the handcrafted parser on the transcribed speech and then corrected by human annotator).

Comments

The previous sections described the general set of dialogue acts in UFAL dialogue systems. However, exact set of dialogue acts depends on a specific application domain and is defined by the domain specific semantic parser.

The only requirement is that all the output of a parser must be accepted by the dialogue manager developed for the particular domain.

Apendix A: UFAL Dialogue acts

Act	Description
`ack()`	back channel – simple OK
`affirm()`	acknowledgement - simple "Yes"
`apology()`	apology for misunderstanding
`bye()`	end of a dialogue
`canthearyou()`	signalling problem with communication channel or that there is an unexpected silence
`confirm(x=y)`	confirm that x equals to y
`iconfirm(x=y)`	implicitly confirm that x equals to y
`deny(x=y)`	denies some information, equivalent to `inform(x != y)`
`hangup()`	end of call because someone hungup
`hello()`	start of a dialogue
`help()`	provide context sensitive help
`inform(x=y)`	inform x equals to y
`inform(name=none)`	inform that "there is no such entity that ... "
`negate()`	negation - simple "No"
`notuderstood()`	informs that the last input was not understood
`null()`	silence, empty sentence, something that is not possible to interpret, does nothing
`repeat()`	asks to repeat the last utterance
`irepeat()`	repeats the last uttered sentence by the system
`reqalts()`	request for alternative options
`reqmore()`	request for more details bout the current option
`request(x)`	request for information about x
`restart()`	restart the dialogue, forget all provided info
`select(x=y)&select(x=z)`	select between two values of the same slot
`silence()`	user or the system does not say anything and remain silent
`thankyou()`	simply thank you

1.1.6 Public Transport Info (English) – data

This directory contains the database used by the English Public Transport Info system, i.e. a list of public transportation stops, number expressions etc. that are understood by the system.

The main database module is located in `database.py`. You may obtain a dump of the database by running `./database.py dump`.

To build all needed generated files that are not versioned, run `build_data.sh`.

1.1.7 Contents of additional data files

Some of the data (for the less populous slots) is included directly in the code `database.py`, but most of the data (e.g., stops and cities) is located in additional list files.

Resources used by public transport direction finders and weather service

The sources of the data that are loaded by the application are:

- `cities.expanded.txt` – list of known cities and towns in the USA. (tab-separated: slot value name + possible forms separated by semicolons; lines starting with '#' are ignored)

- `states.expanded.txt` – list of us state names (same format).
- `stops.expanded.txt` – list of known stop names (same format) in NY.
- `stops.expanded.txt` – list of known stop names (same format) in NY.
- `streets.expanded.txt` – list of known street names (same format)
- `boroughs.expanded.txt` – list of known borough names (same format)
- `cities.locations.csv` – tab separated list of known cities and towns, their state and geo location (longitudellatitude).
- `stops.locations.csv` – tab separated list of stops, their cities and geo location (longitudellatitude).
- `stops.borough.locations.csv` – tab separated list of stops, their boroughs and geo location (longitudellatitude).
- `streets.types.locations.csv` – tab separated list of streets, their boroughs and type (Avenue, Street, Court etc.)

All of these files are generated from `states-in.csv`, `cities-in.csv`, `stops-in.csv`, `streets-in.csv` and `boroughs-in.csv` located at `./preprocessing/resources` using the `expand_states_script.py`, `expand_cities_script.py`, `expand_stops_script.py`, `expand_streets_script.py` and `expand_boroughs_script.py` script respectively. Please note that all forms in `*.expanded.txt` files are lowercased and do not include any punctuation.

Colloquial name variants that are added by hand are located in the `./preprocessing/resources/*-add.txt` files for each slot and are appended to the expansion process.

`build_data.sh` script is combining all the expansion scripts mentioned earlier into one process.

1.1.8 Building a SLU for the PTIen domain

Available data

At this moment, we only have data which were automatically generated using our handcrafted SLU (HDC SLU) parser on the transcribed audio. In general, the quality of the automatic annotation is very good.

The data can be prepared using the `prapare_data.py` script. It assumes that there exist the `indomain_data` directory with links to directories containing `asr_transcribed.xml` files. Then it uses these files to extract transcriptions and generate automatic SLU annotations using the PTIENHDCSLU parser from the `hdc_slu.py` file.

The script generates the following files:

- `*.trn`: contains manual transcriptions
- `*.trn.hdc.sem`: contains automatic annotation from transcriptions using handcrafted SLU
- `*.asr`: contains ASR 1-best results
- `*.asr.hdc.sem`: contains automatic annotation from 1-best ASR using handcrafted SLU
- `*.nbl`: contains ASR N-best results
- `*.nbl.hdc.sem`: contains automatic annotation from n-best ASR using handcrafted SLU

The script accepts `--uniq` parameter for fast generation of unique HDC SLU annotations. This is useful when tuning the HDC SLU.

The script also accepts `--fast` parameter for fast approximate preparation of all data. It approximates the HDC SLU output from an N-best list using output obtained by parsing the 1-best ASR result.

Building the models

First, prepare the data. Link the directories with the in-domain data into the `indomain_data` directory. Then run the following command:

```
./prepare_data.py
```

Second, train and test the models.

```
./train.py && ./test.py && ./test_bootstrap.py
```

Third, look at the `*.score` files or compute the interesting scores by running:

```
./print_scores.sh
```

Future work

- The `prepare_data.py` will have to use ASR, NBLIST, and CONFNET data generated by the latest ASR system instead of the logged ASR results because the ASR can change over time.
- Condition the SLU DialogueActItem decoding on the previous system dialogue act.

Evaluation

Evaluation of ASR from the call logs files

The current ASR performance computed on from the call logs is as follows:

```
Please note that the scoring is implicitly ignoring all non-speech events.

Ref: all.trn
Tst: all.asr
|=============================================================================================|
|            | # Sentences  |  # Words  |   Corr   |   Sub    |   Del    |   Ins    |   Err    |
|-------------------------------------------------------------------------------------------|
| Sum/Avg    |     9111     |   24728   |  56.15   |  16.07   |  27.77   |   1.44   |  45.28   |
|=============================================================================================|
```

The results above were obtained using the Google ASR.

Evaluation of the minimum number of feature counts

Using 9111 training examples, we found that pruning should be set to

- min feature count = 3
- min classifier count = 4

to prevent overfitting.

Cheating experiment: train and test on all data

Due to sparsity issue, the evaluation on proper test and dev sets suffers from sampling errors. Therefore, here we presents results when all data are used as training data and the metrics are evaluated on the training data!!!

Using the `./print_scores.sh` one can get scores for assessing the quality of trained models. The results from experiments are stored in the `old.scores.*` files. Please look at the results marked as `DATA ALL ASR - *`.

If the automatic annotations were correct, we could conclude that the F-measure of the HDC SLU parser on 1-best is higher wne compared to F-measure on N-best%. This is confusing as it looks like that the decoding from n-best lists gives worse results when compared to decoding from 1-best ASR hypothesis.

Evaluation of TRN model on test data

The TRN model is trained on transcriptions and evaluated on transcriptions from test data. Please look at the results marked as `DATA TEST TRN - *`. One can see that the performance of the TRN model on TRN test data is **NOT** 100 % perfect. This is probably due to the mismatch between the train and test data sets. Once more training data will be available, we can expect better results.

Evaluation of ASR model on test data

The ASR model is trained on 1-best ASR output and evaluated on the 1-best ASR output from test data. Please look at the results marked as `DATA TEST ASR - *`. The **ASR model scores significantly better** on the ASR test data when compared to *the HDC SLU parser* when evaluated on the ASR data. The improvement is about 20 % in F-measure (absolute). This shows that SLU trained on the ASR data can be beneficial.

Evaluation of NBL model on test data

The NBL model is trained on N-best ASR output and evaluated on the N-best ASR from test data. Please look at the results marked as `DATA TEST NBL - *`. One can see that using nblists even from Google ASR can help; though only a little (about 1 %). When more data will be available, more test and more feature engineering can be done. However, we are more interested in extracting features from lattices or confusion networks.

Now, we have to wait for a working decoder generating *good* lattices. The OpenJulius decoder is not a suitable as it crashes unexpectedly and therefore it cannot be used in a real system.

1.1.9 Handling non-speech events in Alex

The document describes handling non-speech events in Alex.

ASR

The ASR can generate either:

- a valid utterance
- the ‘‘‘‘ empty sentence word to denote that the input was silence
- the `_noise_` word to denote that the input was some noise or other sound which is not a regular word
- the `_laugh_` word to denote that the input was laugh
- the `_ehm_hmm_` word to denote that the input was ehm or ehm sounds
- the `_inhale_` word to denote that the input was inhale sound
- the `_other_` word to denote that the input was something else that was lost during speech processing approximations such as N-best list enumeration or when the ASR did not provided any result. This is because we do

not know what the input was and it can be both something important or worth ignoring. As such, it deserves special treatment in the system.

SLU

The SLU can generate either:

- a ordinary dialogue act
- the `null()` act which should be ignored by the DM, and the system should respond with `silence()`
- the `silence()` act which denote that the user was silent, a probably reasonable system response is `silence()` as well
- the `other()` act which denote that the input was something else that was lost during processing

The SLU should map:

- "'' to `silence()` - silence will be processed in the DM
- `_noise_`, `_laugh_`, `_ehm_hmm_`, and `_inhale_` to `null()` - noise can be ignored in general
- `_other_` to `other()` - other hypotheses will be handled by the DM, mostly by responding "I did not get that. Can you ... ?"

DM

The DM can generate either:

- a normal dialogue act
- the `silence()` dialogue act

The DM should map:

- `null()` to `silence()` - because the `null()` act denote that the input should be ignored; however there is a problem with this, read the note below for current workaround for this
- `silence()` to `silence()` or a normal dialogue act - the DM should be silent or to ask the user "Are still there?"
- `other()` to `notunderstood()` - to show the user that we did not understood the input and that the input should be rephrased instead of just being repeated.

PROBLEM As of now, both handcrafted and trained SLUs cannot correctly classify the `other()` dialogue act. It has a very low recall for this DA. Instead of the `other()` DA it returns the `null()` DA. Therefore, the `null()` act is processed in DMs as if it was the `other()` DA **for now**.

1.1.10 Public Transport Info, English - telephone service

Description

This application provides information about public transport connections in New York using English language. Just say origin and destination stops and the application will find and tell you about the available connections. You can also specify a departure or arrival time if necessary. It offers bus, tram and metro city connections, and bus and train inter-city connections.

The application is available at the telephone number 1-855-528-7350.

You can also:

- ask for help
- ask for a "restart" of the dialogue and start the conversation again
- end the call - for example, by saying "Good bye."
- ask for repetition of the last sentence
- confirm or reject questions
- ask about the departure or destination station, or confirm it
- ask for the number of transits
- ask for the departure or arrival time
- ask for an alternative connection
- ask for a repetition of the previous connection, the first connection, the second connection, etc.

In addition, the application provides also information about:

- weather forecast
- current time

1.1.11 Public Transport Info, Czech - telephone service

Description

This application provides information about public transport connections in Czech Republic using the Czech language. Just say (in Czech) your origin and destination stops and the application will find and tell you about the available connections. You can also specify a departure or arrival time if necessary. It offers bus, tram and metro city connections, and bus and train inter-city connections.

The application is available at the toll-free telephone number +420 800 899 998.

You can also:

- ask for help
- ask for a "restart" of the dialogue and start the conversation again
- end the call - for example, by saying "Good bye."
- ask for repetition of the last sentence
- confirm or reject questions
- ask about the departure or destination station, or confirm it
- ask for the number of transits
- ask for the departure or arrival time
- ask for an alternative connection
- ask for a repetition of the previous connection, the first connection, the second connection, etc.

In addition, the application provides also information about:

- weather forecast
- the current time

Representation of semantics

Suggestion (MK): It would be better to treat the specification of hours and minutes separately. When they are put together, all ways the whole time expression can be said have to be enumerated in the CLDB manually.

1.1.12 Building of acoustic models using HTK

In this document, we describe building of acoustic models using the HTK toolkit using the provided scripts. These acoustic models can be used with the *OpenJulius* ASR decoder.

We build a different acoustic model for a each language and acoustic condition pair – `LANG_RCOND`. At this time, we provide two sets of scripts for building English and Czech acoustic models using the VOIP data.

In general, the scripts can be described for the language and acoustic condition `LANG_RCOND` as follows:

```
./env_LANG_RCOND.sh          - includes all necessary training parameters: e.g. the train and test da
                               training options including cross word or word internal triphones, lang
./train_LANG_RCOND.sh        - performs the training of acoustic models
./nohup_train_LANG_RCOND.sh  - calls the training script using nohup and redirecting the output into
```

The training process stores some configuration files, the intermediate files, and final models and evaluations in the `model_LANG_RCOND` directory:

```
model_LANG_RCOND/config - config contains the language or recording specific configuration files
model_LANG_RCOND/temp
model_LANG_RCOND/log
model_LANG_RCOND/train
model_LANG_RCOND/test
```

Training models for a new language

Scripts for Czech and English are already created. If you need models for a new language, you can start by copying all the original scripts and renaming them so as to reflect the new language in their name (substitute *_en* or *_cs* with your new language code). You can do this by issuing the following command (we assume `$OLDLANG` is set to either *en* or *cs* and `$NEWLANG` to your new language code):

```
bash htk $ find . -name "*_$OLDLANG*" |
           xargs -n1 bash -c "cp -rvn \$1 \${1/_$OLDLANG/_$NEWLANG}" bash
```

Having done this, references to the new files' names have to be updated, too:

```
bash htk $ find . -name "*_$NEWLANG*" -type f -execdir \
           sed --in-place s/_$OLDLANG/_$NEWLANG/g '{}' \;
```

Furthermore, you need to adjust language-specific resources to the new language in the following ways:

`htk/model_voip_$NEWLANG/monophones0` List all the phones to be recognised, and the special *sil* phone.

`htk/model_voip_$NEWLANG/monophones1` List all the phones to be recognised, and the special *sil* and *sp* phones.

`htk/model_voip_$NEWLANG/tree_ques.hed` Specify phonetic questions to be used for building the decision tree for phone clustering (see [HTKBook], Section 10.5).

`htk/bin/PhoneticTranscriptionCS.pl` You can start from this script or use a custom one. The goal is to implement the orthography-to-phonetics mapping to obtain sequences of phones from transcriptions you have.

htk/common/cmudict.0.7a and htk/common/cmudict.ext This is an alternative approach to the previous point – instead of programming the orthography-to-phonetics mapping, you can list it explicitly in a pronouncing dictionary.

Depending on the way you want to implement the mapping, you want to set `$OLDLANG` to either *cs* or *en*.

To make the scripts work with your new files, you will have to update references to scripts you created. All scripts are stored in the `htk/bin`, `htk/common`, and `htk` directories as immediate children, so you can make the substitutions only in these files.

Credits and the licence

The scripts are based on the HTK Wall Street Journal Training Recipe written by Keith Vertanen (http://www.keithv.com/software/htk/). His code is released under the new BSD licence. The licence note is at http://www.keithv.com/software/htk/. As a result we can re-license the code under the APACHE 2.0 license.

The results

- total training data for voip_en is about 20 hours
- total training data for voip_cs is about 8 hours
- mixtures - there is 16 mixtures is slightly better than 8 mixtures for voip_en
- there is no significant difference in alignment of transcriptions with -t 150 and -t 250
- the Julius ASR performance is about the same as of HDecode
- HDecode works well when cross word phones are trained, however the - performance of HVite decreases significantly
- when only word internal triphones are trained then the HDecode works, - however, its performance is worse than the HVite with a bigram LM
- word internal triphones work well with Julius ASR, do not forget disable CCD (it does not need context handling - though it still uses triphones)
- there is not much gain using the trigram LM in the Caminfo domain (about 1%)

1.1.13 Public Transport Info (Czech) – data

This directory contains the database used by the Czech Public Transport Info system, i.e. a list of public transportation stops, time expressions etc. that are understood by the system.

The main database module is located in `database.py`. You may obtain a dump of the database by running `./database.py dump`.

To build all needed generated files that are not versioned, run `build_data.sh`.

1.1.14 Contents of additional data files

Some of the data (for the less populous slots) is included directly in the code `database.py`, but most of the data (e.g., stops and cities) is located in additional list files.

Resources used by public transport direction finders

The sources of the data that are loaded by the application are:

- `cities.expanded.txt` – list of known cities and towns in the Czech Rep. (tab-separated: slot value name + possible surface forms separated by semicolons; lines starting with '#' are ignored)
- `stops.expanded.txt` – list of known stop names (same format)
- `cities_stops.tsv` – "compatibility table": lists compatible city-stops pairs, one entry per line (city and stop are separated by tabs). Only the primary stop and city names are used here.

The files `cities.expanded.txt` and `stops.expanded.txt` are generated from `cities.txt` and `stops.txt` using the `expand_stops.py` script (see documentation in the file itself; you need to have Morphodita Python bindings installed to successfully run this script). Please note that the surface forms in them are lowercased and do not include any punctuation (this can be obtained by setting the `-l` and `-p` parameters of the `expand_stops.py` script).

Colloquial stop names' variants that are added by hand are located in the `stops-add.txt` file and are appended to the `stops.txt` before performing the expansion.

Additional resources for the CRWS/IDOS directions finder

Since the CRWS/IDOS directions finder uses abbreviated stop names that need to be spelled out in ALEX, there is an additional resource file loaded by the system:

- `idos_map.tsv` – a mapping from the slot value names (city + stop) to abbreviated CRWS/IDOS names (stop list + stop)

The `convert_idos_stops.py` script is used to expand all possible abreviations and produce a mapping from/to the original CRWS/IDOS stop names as they appear, e.g., at the IDOS portal .

Resources used by the weather information service

The weather service uses one additional file:

- `cities_locations.tsv` – this file contains GPS locations of all cities in the Czech Republic.

1.1.15 Building a SLU for the PTIcs domain

Available data

At this moment, we only have data which were automatically generated using our handcrafted SLU (HDC SLU) parser on the transcribed audio. In general, the quality of the automatic annotation is very good.

The data can be prepared using the `prapare_data.py` script. It assumes that there exist the `indomain_data` directory with links to directories containing `asr_transcribed.xml` files. Then it uses these files to extract transcriptions and generate automatic SLU annotations using the PTICSHDCSLU parser from the `hdc_slu.py` file.

The script generates the following files:

- `*.trn`: contains manual transcriptions
- `*.trn.hdc.sem`: contains automatic annotation from transcriptions using handcrafted SLU
- `*.asr`: contains ASR 1-best results
- `*.asr.hdc.sem`: contains automatic annotation from 1-best ASR using handcrafted SLU

- `*.nbl`: contains ASR N-best results
- `*.nbl.hdc.sem`: contains automatic annotation from n-best ASR using handcrafted SLU

The script accepts `--uniq` parameter for fast generation of unique HDC SLU annotations. This is useful when tuning the HDC SLU.

Building the DAILogRegClassifier models

First, prepare the data. Link the directories with the in-domain data into the `indomain_data` directory. Then run the following command:

```
./prepare_data.py
```

Second, train and test the models.

:: cd ./dailogregclassifier

    ./train.py && ./test_trn.py && ./test_hdc.py && ./test_bootstrap_trn.py && ./test_bootsrap_hdc.py

Third, look at the `*.score` files or compute the interesting scores by running:

```
./print_scores.sh
```

Future work

- Exploit ASR Lattices instead of long NBLists.
- Condition the SLU DialogueActItem decoding on the previous system dialogue act.

Evaluation

Evaluation of ASR from the call logs files

The current ASR performance computed on from the call logs is as follows:

```
Please note that the scoring is implicitly ignoring all non-speech events.

Ref: all.trn
Tst: all.asr
|=============================================================================================|
|            | # Sentences  |  # Words  |   Corr   |   Sub    |   Del    |   Ins    |   Err    |
|-------------------------------------------------------------------------------------------|
| Sum/Avg    |     9111     |   24728   |  56.15   |  16.07   |  27.77   |   1.44   |  45.28   |
|=============================================================================================|
```

The results above were obtained using the Google ASR.

Evaluation of the minimum number of feature counts

Using 9111 training examples, we found that pruning should be set to

- min feature count = 3
- min classifier count = 4

to prevent overfitting.

Cheating experiment: train and test on all data

Due to sparsity issue, the evaluation on proper test and dev sets suffers from sampling errors. Therefore, here we presents results when all data are used as training data and the metrics are evaluated on the training data!!!

Using the `./print_scores.sh` one can get scores for assessing the quality of trained models. The results from experiments are stored in the `old.scores.*` files. Please look at the results marked as `DATA ALL ASR - *`.

If the automatic annotations were correct, we could conclude that the F-measure of the HDC SLU parser on 1-best is higher wne compared to F-measure on N-best%. This is confusing as it looks like that the decoding from n-best lists gives worse results when compared to decoding from 1-best ASR hypothesis.

Evaluation of TRN model on test data

The TRN model is trained on transcriptions and evaluated on transcriptions from test data. Please look at the results marked as `DATA TEST TRN - *`. One can see that the performance of the TRN model on TRN test data is **NOT** 100 % perfect. This is probably due to the mismatch between the train and test data sets. Once more training data will be available, we can expect better results.

Evaluation of ASR model on test data

The ASR model is trained on 1-best ASR output and evaluated on the 1-best ASR output from test data. Please look at the results marked as `DATA TEST ASR - *`. The **ASR model scores significantly better** on the ASR test data when compared to *the HDC SLU parser* when evaluated on the ASR data. The improvement is about 20 % in F-measure (absolute). This shows that SLU trained on the ASR data can be beneficial.

Evaluation of NBL model on test data

The NBL model is trained on N-best ASR output and evaluated on the N-best ASR from test data. Please look at the results marked as `DATA TEST NBL - *`. One can see that using nblists even from Google ASR can help; though only a little (about 1 %). When more data will be available, more test and more feature engineering can be done. However, we are more interested in extracting features from lattices or confusion networks.

Now, we have to wait for a working decoder generating *good* lattices. The OpenJulius decoder is not a suitable as it crashes unexpectedly and therefore it cannot be used in a real system.

1.1.16 Interective tests and unit tests

Testing of Alex can be divided into interactive tests, which depends on on some activity of a user e.g. calling a specific phone number or listening to some audi file, and unit tests, which are testing some very specific properties of algorithms or libraries.

Interactive tests

This directory contains only (interactive) tests, which can't be automated and the results must be verified by humans! E.g. playing or recording audio, testing VOIP connections.

Unit tests

Note that the unit tests should be placed in the same directory as the tested module and the name should be `test_*.py` e.g. `test_module_name.py`.

Using unittest module:

```
$ python -m unittest alex.test.test_string
```

This approach works everywhere but doesn't support test discovery.

Using nose test discovery framework, testing can largely automated. Nose searchs through packages and runs every test. Tests must be named `test_<something>.py` and must not be executable. Tests doesn't have to be run from project root, nose is able to find project root on its own.

How should my unit tests look like?

- Use unittest module
- Name the test file `test_<something>.py`
- Make the test file not executable

1.1.17 RepeatAfterMe (RAM) for Czech - speech data collection

This application is useful for bootstraping of speech data. It asks the caller to repeat sentences which are randomly sampled from a set of preselected sentences.

- The Czech sentences (`sentences_es.txt`) are from Karel Capek novels Matka and RUR, and the Prague's Dependency Treebank.
- The Spanish sentences (`sentences_es.txt`) are taken from the Internet

If you want to run `ram_hub.py` on some specific phone number than specify the appropriate extension config:

```
$ ./ram_hub.py -c ram_hub_LANG.cfg  ../../resources/private/ext-PHONENUMBER.cfg
```

After collection desired number of calls, use `copy_wavs_for_transcription.py` to extract the wave files from the `call_logs` subdirectory for transcription. The files will be copied into into `RAM-WAVs` directory.

These calls must be transcribed by the Transcriber or some similar software.

1.1.18 Approach to bootstraping the domain specific language models

```
**WARNING**: Please note that domain specific language models are build in ./alex/applications/*/lm
This text explains a simple approach to building a domain specific language models, which can be dif
domain.
```

While an acoustic model can be build domain independent, the language models (LMs) must be domain specific to ensure high accuracy of the ASR.

In general, building an in-domain LM is easy as long as one has enough of in-domain training data. However, when the in-domain data is scarce, e.g. when deploying a new dialogue system, this task is difficult and there is a need for some bootstrap solution.

The approach described here builds on:

1. some bootstrap text - probably handcrafted, which captures the main aspects of the domain
2. LM classes - which clusters words into classes, this can be derived from some domain ontology. For example, all food types belong to the FOOD class and all public transport stops stops belong to the STOP class

3. in-domain data - collected using some prototype or final system
4. general out-of-domain data - for example Wikipedia - from which is selected a subset of data, similar to our in-domain data

Then a simple process of building a domain specific language model can described as follows:

1. Append bootstrap text to the text extracted from the indomain data.
2. Build a class based language model using the data generated in the previous step and the classes derived from the domain ontology.
3. Score the general (domain independent) data using the LM build in the previous step.
4. Select some sentences with the lowest perplexity given the class based language model.
5. Append the selected sentences to the training data generated in the 1. step.
6. Re-build the class based language model.
7. Generate dictionaries.

Data for building general LMs

To get free general out-of-domain text data, we use the free W2C – Web to Corpus – Corpora available from the LINDAT project at: https://ufal-point.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0022-6133-9

- English: https://ufal-point.mff.cuni.cz/repository/xmlui/bitstream/handle/11858/00-097C-0000-0022-6133-9/eng.txt.gz
- Czech: https://ufal-point.mff.cuni.cz/repository/xmlui/bitstream/handle/11858/00-097C-0000-0022-6133-9/ces.txt.gz

Structure of each domain scripts

Each of the projects should contain:

1. build.py - builds the final LMs, and computes perplexity of final LMs

Necessary files for the LM

For each domain the LM package should contain:

1. ARPA trigram language model (`final.tg.arpa`)
2. ARPA bigram language model (`final.bg.arpa`)
3. HTK wordnet bigram language model (`final.bg.wdnet`)
4. List of all words in the language model (`final.vocab`)
5. Dictionary including all words in the language model using compatible phone set with the language specific acoustic model (`final.dict` - without pauses and `final.dict.sp_sil` with short and long pauses)

CamInfoRest

For more details please see `alex.applications.CamInfoRest.lm.README`.

PTIcs

For more details please see *Building the language model for the Public Transport Info telephone service (Czech)*.

1.1.19 Online distribution of resource files such as ASR, SLU, NLG models

Large binary files are difficult to store in git. Therefore, files such as resource files for ASR, SLU or NLG are distributed online and on-demand.

To use this functionality you have to use the `online_update(file_name)` function from the `alex.utils.config` package. The functions checks the file name whether it exists locally and it is up-to-date. If it is missing or it is old, then a new version from the server is downloaded.

The function returns name if the downloaded file which equal to input file name. As a result it is transparent in a way, that this function can be used everywhere a file name must be entered.

The server is set to `https://vystadial.ms.mff.cuni.cz/download/`; however, it can be changed using the `set_online_update_server(server_name)` function from inside a config file, e.g. the (first) default config file.

1.1.20 Building of acoustic models using KALDI

In this document, we describe building of acoustic models using the KALDI toolkit and the provided scripts. These acoustic models can be used with the *Kaldi* decoders and especially with the Python wrapper of `LatgenFasterDecoder` which is integrated with Alex.

We build a different acoustic model for a each language and acoustic condition pair – `LANG_RCOND`. At this time, we provide two sets of scripts for building English and Czech acoustic models using the VOIP data.

In general, the scripts can be described for the language and acoustic condition `LANG_RCOND` as follows:

Summary

- Requires KALDI installation and Linux environment. (Tested on Ubuntu 10.04, 12.04 and 12.10.) Note: We recommend Kaldi fork Pykaldi, because you will need it also for integrated Kaldi decoder to Alex.
- Recipes deployed with the Kaldi toolkit are located at `$KALDI_ROOT/egs/name_of_recipe/s[1-5]/`. This recipe requires to set up `$KALDI_ROOT` variable so it can use Kaldi binaries and scripts from `$KALDI_ROOT/egs/wsj/s5/`.

Details

- The recommended settings are stored at `env_LANG_RCONG.sh` e.g `env_voip_en.sh`
- We recommend to adjust the settings in file *env_LANG_RCONG_CUSTOM.sh'* e.g. `env_voip_en_CUSTOM.sh`. See below. Do not commit this file to the git repository!
- Our scripts prepare the data to the expected format to `$WORK` directory.
- Experiment files are stored to `$EXP` directory.
- The symbolic links to `$KALDI_ROOT/wsj/s5/utils` and `$KALDI_ROOT/wsj/s5/steps` are automatically created.
- The files `path.sh`, `cmd.sh` are necessary to `utils` and `steps` scripts. Do not relocate them!

- Language model (LM) is either built from the training data using SRILM or specified in `env_LANG_RCOND.sh`.

Example of `env_voip_en_CUSTOM.sh`

```
# uses every utterance for the recipe every_N=10 is nice for debugging
export EVERY_N=1
# path to built Kaldi library and scripts
export KALDI_ROOT=/net/projects/vystadial/lib/kronos/pykaldi/kaldi

export DATA_ROOT=/net/projects/vystadial/data/asr/cs/voip/
export LM_paths="build0 $DATA_ROOT/arpa_bigram"
export LM_names="build0 vystadialbigram"

export CUDA_VISIBLE_DEVICES=0  # only card 0 (Tesla on Kronos) will be used for DNN training
```

Running experiments

Before running the experiments, check that:

- you have the Kaldi toolkit compiled: - http://github.com/UFAL-DSG/pykaldi (Recommended Kaldi fork, tested, necessary for further Alex integration) - http://sourceforge.net/projects/kaldi/ (alternative, main Kaldi repository) - In order to compile Kaldi we suggest:

```
# build openfst
pushd kaldi/tools
make openfst_tgt
popd

# download ATLAS headers
pushd kaldi/tools
make atlas
popd

# generate Kaldi makefile ``kaldi.mk`` and compile Kaldi
pushd kaldi/src
./configure
make && make test
popd
```

- you have SRILM compiled. (This is needed for building a language model) unless you supply your own LM in the ARPA format.)

```
pushd kaldi/tools
# download the srilm.tgz archive from http://www.speech.sri.com/projects/srilm/download.html
./install_srilm.sh
pushd
```

- the `train_LANG_RCOND` script will see the Kaldi scripts and binaries. Check for example that `$KALDI_ROOT/egs/wsj/s5/utils/parse_options.sh` is valid path.
- in `cmd.sh`, you switched to run the training on a SGE[*] grid if required (disabled by default) and `njobs` is less than number of your CPU cores.

Start the recipe by running `bash train_LANG_RCOND.sh`.

Extracting the results and trained models

The main script, `bash train_LANG_RCOND.sh`, performs not only training of the acoustic models, but also decoding. The acoustic models are evaluated during running the scripts and evaluation reports are printed to the standard output.

The `local/results.py exp` command extracts the results from the `$EXP` directory. It is invoked at the end of the `train_LANG_RCOND.sh` script.

If you want to use the trained acoustic model outside the prepared script, you need to build the `HCLG` decoding graph yourself. (See http://kaldi.sourceforge.net/graph.html for general introduction to the FST framework in Kaldi.) The `HCLG.fst` decoding graph is created by `utils/mkgraph.sh`. See `run.sh` for details.

Credits and license

The scripts were based on Voxforge KALDI recipe http://vpanayotov.blogspot.cz/2012/07/voxforge-scripts-for-kaldi.html . The original scripts as well as theses scripts are licensed under APACHE 2.0 license.

1.1.21 Building a voice activity detector (VAD)

This text described how to build a voice activity detector (VAD) for Alex. This work builds multilingual VAD. That means that we do not have VADs for individual languages but rather only one. It appears that NN VAD has the capacity to distinguish between non-speech and speech in any language.

As of now, we use VAD based on neural networks (NNs) implemented in the Theano toolkit. The main advantage that the same code can efficiently run both CPUs and GPUs and Theano implements automatic derivations. Automatic derivations is very useful especially when gradient descend techniques, such as stochastic gradient descent, are used for model parameters optimisation.

Old GMM code is still present but it may not work and its performance would be significantly worse that of the current NN implementation.

Experiments and the notes for the NN VAD

- testing is performed on randomly sampled data points (20%) from the entire set
- L2 regularisation must be very small, in addition it does not help much
- instead of MFCC, we use mel-filter banks coefficients only. It looks like the performance is the same or even better
- as of 2014-09-19 the best compromise between the model complexity and the performance appears to be.
  - 30 previous frames
  - 15 next frames
  - 512 hidden units
  - 4 hidden layers
  - tanh hidden layer activation
  - 4x amplification of the central frame compared to outer frames
  - discriminative pre-training
  - given this setup we get about 95.3 % frame accuracy on about 27 million of all data

Data

```
data_vad_sil     # a directory with only silence, noise data and its mlf file
data_voip_cs     # a directory where CS data reside and its MLF (phoneme alignment)
data_voip_en     # a directory where EN data reside and its MLF (phoneme alignment)
model_voip       # a directory where all the resulting models are stored.
```

Scripts

```
upload_models.sh                      # uploads all available models in ``model_voip`` onto the Alex
train_voip_nn_theano_sds_mfcc.py      # this is the main trainign script, see its help for more detai
bulk_train_nn_theano_mbo_31M_sgd.sh   # script with curently ``optimal`` setting for VAD
```

Comments

To save some time especially for multiple experiments on the same data, we store preprocessed speech parametrisation. The speech parametrisation is stored because it takes about 7 hours to produce. However, it takes only 1 minute to load from a disk file. The `model_voip` directory stores this speech parametrisation in `*.npc` files. There fore if new data is added, then these NPC files must be deleted. If there are no NPC files then they are automatically generated from the available WAV files.

The `data_voip_{cs,en}` alignment files (mlf files) can be trained using scripts `alex/alex/tools/htk` or `alex/alex/tools/kaldi`. See the `train_voip_{cs,en}.sh` scripts in one of the directories. Note that the Kaldi scripts first store alignment in `ctm` format and later converts it to `mlf` format.

1.1.22 Utils for building decoding graph HCLG

Summary

The `build_hclg.sh` script formats language model (LM) and acoustic model (AM) into files (e.g. HCLG) formated for Kaldi decoders.

The scripts extracts phone lists and sets from lexicon given the acoustic model (AM), the phonetic decision tree (tree) and the phonetic dictionary(lexicon).

The script silently supposes the same phone lists are generated from lexicon as the these used for training AM. If they are not the same, the script crashes.

The use case. Run the script with trained AM on full phonetic set for given language, pass the script also the tree used for tying the phonetic set and also give the script your LM and corresponding lexicon. The lexicon and the LM should also cover the full phonetic set for given language.

The `decode_indomain.py` script uses `HCLG.fst` and the rest of files generated by `build_hclg.sh` and performes decoding on prerecorded wav files. The reference speech transcription and path to the wav files are extracted from collected call logs. The wav files should be from one domain and the LM used to build `HCLG.fst` should be from the same domain. The `decode_indomain.py` also evaluates the decoded transcriptions. The Word Error Rate (WER), Real Time Factor (RTF) and other minor statistics are collected.

Dependencies of build_hclg.sh

The build_hclg.sh script requires the scripts listed belofw from `$KALDI_ROOT/egs/wsj/s5/utils`. The "utils scripts transitevely uses scripts from `$KALDI_ROOT/egs/wsj/s5/steps`. The dependency is solved in `path.sh` script which create corresponding symlinks and adds Kaldi binaries to your system path.

You just needed to set up `KALDI_ROOT` root variable and provide correct arguments. Try to run

Needed scripts from `utils` symlinked directory. * gen_topo.pl * add_lex_disambig.pl * apply_map.pl * eps2disambig.pl * find_arpa_oovs.pl * gen_topo.pl * make_lexicon_fst.pl * remove_oovs.pl * s2eps.pl * sym2int.pl * validate_dict_dir.pl * validate_lang.pl * parse_options.sh

Scripts from the list use Kaldi binaries, so you need Kaldi compiled on your system. The script `path.sh` adds Kaldi binaries to the `PATH` and also creates symlinks to `utils` and `steps` directories, where the helper scripts are located. You only need to set up `$KALDI_ROOT` variable.

1.1.23 Public Transport Info, English - telephone service

Running the system at UFAL with the full UFAL access

There are multiple configuration that can used to run the system. In general, it depends on what components you want go use and on what telephone extension you want to run the system.

Within UFAL, we run the system using the following commands:

- `vhub_mta1` - deployment of our live system on a 1-855-528-7350 phone number, with the default configuration
- `vhub_mta2` - a system deployed to backup the system above
- `vhub_mta3` - a system deployed to backup the system above
- `vhub_mta_btn` - a system deployed to backup the system above accessible via web page http://alex-ptien.com

To test the system we use:

- `vhub_devel` - default devel version of our system deployed on our test extension, logging locally into `../call_logs`

Running the system without the full UFAL access

Users outside UFAL can run the system using the following commands:

- `vhub_private_ext_google_only_hdc_slu` - default version of our system deployed on private extension specified in `private_ext.cfg`, using HDC_SLU, Google ASR, TTS, Directions, logging locally into `../call_logs`
- `vhub_private_ext_google_kaldi_hdc_slu` - default version of our system deployed on private extension specified in `private_ext.cfg`, using HDC_SLU, Google TTS, Directions, and KALDI ASR, logging locally into `../call_logs`

If you want to test the system on your private extension, then modify the `private_ext.cfg` config. You must set your SIP domain including the port, user login, and password. Please make sure that you do not commit your login information into the repository.

```
config = {
        'VoipIO': {
                # default testing extesion
                'domain':   "*:5066",
                'user':     "*",
                'password': "*",
        },
}
```

Also, you will have to create a "private" directory where you can store your private configurations. As the private default configuration is not part of the Git repository, please make your own empty version of the private default configuration as follows.

```
mkdir alex/resources/private
echo "config = {}" > alex/resources/private/default.cfg
```

CHAPTER 2

Alex modules

2.1 alex package

2.1.1 Subpackages

alex.applications package

Subpackages

alex.applications.PublicTransportInfoCS package

Subpackages

alex.applications.PublicTransportInfoCS.data package

Submodules

alex.applications.PublicTransportInfoCS.data.add_cities_to_stops module A script that creates a compatibility table from a list of stops in a certain city and its neighborhood and a list of towns and cities.

Usage:

./add_cities_to_stops.py [-d "Main city"] stops.txt cities.txt cities_stops.tsv

alex.applications.PublicTransportInfoCS.data.add_cities_to_stops.**add_cities_to_stops**(*cities*, *stops*, *main_ci*

alex.applications.PublicTransportInfoCS.data.add_cities_to_stops.**get_city_for_stop**(*cities*, *stop*, *main_city*)

alex.applications.PublicTransportInfoCS.data.add_cities_to_stops.**load_list**(*filename*, *suppress_comments=False*, *cols=1*)

alex.applications.PublicTransportInfoCS.data.add_cities_to_stops.**main**()

alex.applications.PublicTransportInfoCS.data.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoCS.data.convert_idos_stops module

alex.applications.PublicTransportInfoCS.data.database module

alex.applications.PublicTransportInfoCS.data.download_data module

alex.applications.PublicTransportInfoCS.data.expand_stops module

alex.applications.PublicTransportInfoCS.data.get_cities_location module A script that collects the locations of all the given cities using the Google Geocoding API.

Usage:

./get_cities_locations.py [-d delay] [-l limit] [-a] cities_locations-in.tsv cities_locations-out.tsv

-d = delay between requests in seconds (will be extended by a random period up to 1/2 of the original value)

-l = limit maximum number of requests -a = retrieve all locations, even if they are set

alex.applications.PublicTransportInfoCS.data.get_cities_location.**get_google_coords**(*city*)
Retrieve (all possible) coordinates of a city using the Google Geocoding API.

alex.applications.PublicTransportInfoCS.data.get_cities_location.**random**() → x in the interval [0, 1).

alex.applications.PublicTransportInfoCS.data.ontology module

alex.applications.PublicTransportInfoCS.data.ontology.**add_slot_values_from_database**(*slot*, *category*, *exceptions=set*

alex.applications.PublicTransportInfoCS.data.ontology.**load_additional_information**(*fname*, *slot*, *keys*)

alex.applications.PublicTransportInfoCS.data.ontology.**load_compatible_values**(*fname*, *slot1*, *slot2*)

Module contents

alex.applications.PublicTransportInfoCS.hclg package

Submodules

alex.applications.PublicTransportInfoCS.hclg.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoCS.hclg.kaldi_calibration module

Module contents

alex.applications.PublicTransportInfoCS.slu package

Subpackages

alex.applications.PublicTransportInfoCS.slu.dailogregclassifier package

Submodules

alex.applications.PublicTransportInfoCS.slu.dailogregclassifier.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoCS.slu.dailogregclassifier.download_models module

alex.applications.PublicTransportInfoCS.slu.dailogregclassifier.test_bootstrap_trn module

alex.applications.PublicTransportInfoCS.slu.dailogregclassifier.test_trn module

alex.applications.PublicTransportInfoCS.slu.dailogregclassifier.train module

Module contents

alex.applications.PublicTransportInfoCS.slu.dainnclassifier package

Submodules

alex.applications.PublicTransportInfoCS.slu.dainnclassifier.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoCS.slu.dainnclassifier.download_models module

alex.applications.PublicTransportInfoCS.slu.dainnclassifier.test_bootstrap_trn module

alex.applications.PublicTransportInfoCS.slu.dainnclassifier.test_trn module

Module contents

Submodules

alex.applications.PublicTransportInfoCS.slu.add_to_bootstrap module A simple script for adding new utterances along with their semantics to bootstrap.sem and bootstrap.trn.

Usage:

./add_to_bootsrap < input.tsv

The script expects input with tab-separated transcriptions + semantics (one utterance per line). It automatically generates the dummy 'bootstrap_XXXX.wav' identifiers and separates the transcription and semantics into two files.

`alex.applications.PublicTransportInfoCS.slu.add_to_bootstrap.`**`main`**`()`

alex.applications.PublicTransportInfoCS.slu.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoCS.slu.consolidate_keyfiles module This scripts consolidates all input key files. That means, that it generates new keyfiles ({old_name}.pruned, which contains only entries common to all input ket files.

`alex.applications.PublicTransportInfoCS.slu.consolidate_keyfiles.`**`main`**`()`

alex.applications.PublicTransportInfoCS.slu.gen_bootstrap module

alex.applications.PublicTransportInfoCS.slu.gen_uniq module

alex.applications.PublicTransportInfoCS.slu.prepare_data module

alex.applications.PublicTransportInfoCS.slu.prepare_hdc_sem_from_trn module

alex.applications.PublicTransportInfoCS.slu.test_bootstrap_hdc module

alex.applications.PublicTransportInfoCS.slu.test_hdc module

alex.applications.PublicTransportInfoCS.slu.test_hdc_utt_dict module

Module contents

Submodules

alex.applications.PublicTransportInfoCS.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoCS.crws_enums module Various enums, semi-automatically adapted from the CHAPS CRWS enum list written in C#.

Comments come originally from the CRWS description and are in Czech.

alex.applications.PublicTransportInfoCS.crws_enums.**BEDS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**CLIENTEXCEPTION_CODE**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**COMBFLAGS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**COOR**
alias of Enum

class alex.applications.PublicTransportInfoCS.crws_enums.**CRCONST**

DELAY_CD = 'CD:'

DELAY_INTERN = 'X{0}_{1}:'

DELAY_INTERN_EXT = 'Y{0}_{1}:'

DELAY_TELMAX1 = 'TELMAX1:'

DELAY_ZSR = 'ZSR:'

EXCEPTIONEXCLUSION_CD = 'CD:'

alex.applications.PublicTransportInfoCS.crws_enums.**DELTAMAX**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**DEP_TABLE**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**EXFUNCTIONRESULT**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**FCS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**LISTID**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**OBJECT_STATUS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**REG**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**REMMASK**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**ROUTE_FLAGS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**SEARCHMODE**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**ST**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**SVCSTATE**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TIMETABLE_FLAGS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TRCAT**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TRSUBCAT**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TTDETAILS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TTERR**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TTGP**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TTINFODETAILS**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**TTLANG**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**VF**
alias of Enum

alex.applications.PublicTransportInfoCS.crws_enums.**enum**(***enums*)

alex.applications.PublicTransportInfoCS.cs_morpho module

alex.applications.PublicTransportInfoCS.directions module

alex.applications.PublicTransportInfoCS.exceptions module

exception alex.applications.PublicTransportInfoCS.exceptions.**PTICSHDCPolicyException**
Bases: alex.components.dm.exceptions.DialoguePolicyException

alex.applications.PublicTransportInfoCS.hdc_policy module

alex.applications.PublicTransportInfoCS.hdc_slu module

alex.applications.PublicTransportInfoCS.platform_info module

class alex.applications.PublicTransportInfoCS.platform_info.**CRWSPlatformInfo**(*crws_response*, *finder*)

Bases: object

find_platform_by_station(*to_obj*)

find_platform_by_train_name(*train_name*)

station_name_splitter = <_sre.SRE_Pattern object at 0x7ff392b58d40>

class alex.applications.PublicTransportInfoCS.platform_info.**PlatformFinderResult**(*platform*, *track*, *direction*)

Bases: object

class alex.applications.PublicTransportInfoCS.platform_info.**PlatformInfo**(*from_stop*, *to_stop*, *from_city*, *to_city*, *train_name*, *directions*)

Bases: object

alex.applications.PublicTransportInfoCS.platform_info_test module

class alex.applications.PublicTransportInfoCS.platform_info_test.**PlatformInfoTest**(*methodName='*

Bases: unittest.case.TestCase

test_matching()

alex.applications.PublicTransportInfoCS.preprocessing module

alex.applications.PublicTransportInfoCS.test_hdc_policy module

alex.applications.PublicTransportInfoCS.test_hdc_slu module

Module contents

alex.applications.PublicTransportInfoEN package

Subpackages

alex.applications.PublicTransportInfoEN.data package

Subpackages

alex.applications.PublicTransportInfoEN.data.preprocessing package

Submodules

alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_script_manual module A script that basically creates a csv file that contains a list of places from INPUT_FILE sith second column of a STRING_SAME_FOR_ALL and the benefit is that it can merge with already existing OUTPUT_FILE

unless -c flag is set.

Usage: /.compatibility_script_manual –name OUTPUT_FILE –main-place STRING_SAME_FOR_ALL –list INPUT_FILE [-c]

alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_script_manual.**hand**

alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_script_manual.**main**

alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_script_manual.**read**

alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_script_manual.**save**

alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_script_manual.**stic**

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv module A script that takes mta stops file and it selects important fields and saves them (works with GTFS mainly) Usage:

./mta_to_csv.py [-m: main_city] [-o: output_file] stops.txt

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**average_same_stops**(*sa*

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**extract_fields**(*lines*, *header*, *main_ci* *skip_co*

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**get_column_index**(*head* *caption*, *default*

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**group_by_name**(*data*)

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**load_list**(*filename*, *skip_comments*

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**main**()

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**remove_duplicities**(*li*

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**remove_following_dupl**

alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv.**write_data**(*file_name*, *data*)

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment module A script that takes mta stops, it splits them by special characters and each item takes for a street

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**aver**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**extr**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**get_**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**grou**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**load**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**main**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**remo**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**remo**

alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment.**writ**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv module A script that takes us cities (city state_code)file and state-codes and it joins them

Usage:

./us_cities_to_csv.py [-o: output_file] cities.txt state-codes.txt

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**average_same_c**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**extract_fields**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**get_column_inde**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**group_by_city_a**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**load_list**(*filenam*
skip_c

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**load_state_code**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**main**()

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**remove_duplicit**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**remove_followi**

alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv.**write_data**(*file_*
data)

Module contents

Submodules

alex.applications.PublicTransportInfoEN.data.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoEN.data.database module

alex.applications.PublicTransportInfoEN.data.download_data module

alex.applications.PublicTransportInfoEN.data.expand_boroughs_script module A script that creates an expansion from a preprocessed list of boroughs

For usage write expand_boroughs_script.py -h

alex.applications.PublicTransportInfoEN.data.expand_boroughs_script.**all_to_lower**(*site_list*)

alex.applications.PublicTransportInfoEN.data.expand_boroughs_script.**handle_boroughs**(*boroughs_*
bor-
oughs_ou
bor-
oughs_ap
no_cache

alex.applications.PublicTransportInfoEN.data.expand_boroughs_script.**main**()

alex.applications.PublicTransportInfoEN.data.expand_cities_script module A script that creates an expansion from a preprocessed list of cities

For usage write expand_cities_script.py -h

alex.applications.PublicTransportInfoEN.data.expand_cities_script.**all_to_lower**(*site_list*)

alex.applications.PublicTransportInfoEN.data.expand_cities_script.**handle_cities**(*cities_in*,
cities_out,
cities_append,
no_cache=Fals

alex.applications.PublicTransportInfoEN.data.expand_cities_script.**main**()

alex.applications.PublicTransportInfoEN.data.expand_states_script module A script that creates an expansion from a preprocessed list of states

For usage write expand_states_script.py -h

alex.applications.PublicTransportInfoEN.data.expand_states_script.**handle_states**(*states_in*, *states_out*, *states_append*, *no_cache=Fals*

alex.applications.PublicTransportInfoEN.data.expand_states_script.**main**()

alex.applications.PublicTransportInfoEN.data.expand_stops_script module A script that creates an expansion from a list of stops

For usage write expand_stops_script.py -h

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**append**(*major*, *minor*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**expand_place**(*stop_list*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**file_check**(*filename*, *message=u'reading file'*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**get_column_index**(*header*, *caption*, *default*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**hack_stops**(*stops*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**handle_compatibility**(*file_in*, *file_ou*, *no_ca*

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**handle_csv**(*csv_in*, *csv_out*, *no_cache=False*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**load_list**(*filename*, *skip_comments=True*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**main**()

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**merge**(*primary*, *secondary*, *surpress_warning=True*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**preprocess_line**(*line*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**process_places**(*places_in*, *place_out*, *places_add*, *no_cache=Fals*

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**read_compatibility**(*filename*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**read_expansions**(*stops_expande*

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**read_exports**(*filename*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**read_first_column**(*filename*, *sur-*
press_warn

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**read_two_columns**(*filename*)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**save_list**(*output_file*, *out-*
put_list)

alex.applications.PublicTransportInfoEN.data.expand_stops_script.**save_out**(*output_file*, *out-*
put_dict, *sep-*
a-
ra-
tor=u';
')

alex.applications.PublicTransportInfoEN.data.expand_streets_script module A script that creates an expansion from a list of stops

For usage write expand_stops_script.py -h

alex.applications.PublicTransportInfoEN.data.expand_streets_script.**main**()

alex.applications.PublicTransportInfoEN.data.ontology module

alex.applications.PublicTransportInfoEN.data.ontology.**add_slot_values_from_database**(*slot*, *cat-*
e-
gory, *ex-*
cep-
tions=set

alex.applications.PublicTransportInfoEN.data.ontology.**load_compatible_values**(*fname*, *slot1*, *slot2*)

alex.applications.PublicTransportInfoEN.data.ontology.**load_geo_values**(*fname*, *slot1*, *slot2*, *sur-*
press_warning=True)

alex.applications.PublicTransportInfoEN.data.ontology.**load_street_type_values**(*fname*, *sur-*
press_warning=Fa

Module contents

alex.applications.PublicTransportInfoEN.slu package

Submodules

alex.applications.PublicTransportInfoEN.slu.add_to_bootstrap module A simple script for adding new utterances along with their semantics to bootstrap.sem and bootstrap.trn.

Usage:

./add_to_bootsrap < input.tsv

The script expects input with tab-separated transcriptions + semantics (one utterance per line). It automatically generates the dummy 'bootstrap_XXXX.wav' identifiers and separates the transcription and semantics into two files.

`alex.applications.PublicTransportInfoEN.slu.add_to_bootstrap.`**`main`**`()`

alex.applications.PublicTransportInfoEN.slu.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

    import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

    pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoEN.slu.consolidate_keyfiles module
`alex.applications.PublicTransportInfoEN.slu.consolidate_keyfiles.`**`main`**`()`

alex.applications.PublicTransportInfoEN.slu.gen_bootstrap module

alex.applications.PublicTransportInfoEN.slu.prepare_data module

alex.applications.PublicTransportInfoEN.slu.query_google module
`alex.applications.PublicTransportInfoEN.slu.query_google.`**`main`**`()`

alex.applications.PublicTransportInfoEN.slu.test_bootstrap module

Module contents

Submodules

alex.applications.PublicTransportInfoEN.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

    import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.PublicTransportInfoEN.directions module

class alex.applications.PublicTransportInfoEN.directions.**Directions**(***kwargs*)
Bases: alex.applications.PublicTransportInfoEN.directions.Travel

Ancestor class for transit directions, consisting of several routes.

class alex.applications.PublicTransportInfoEN.directions.**DirectionsFinder**
Bases: object

Abstract ancestor for transit direction finders.

get_directions(*from_city*, *from_stop*, *to_city*, *to_stop*, *departure_time=None*, *arrival_time=None*, *parameters=None*)
Retrieve the transit directions from the given stop to the given stop at the given time.

Should be implemented in derived classes.

class alex.applications.PublicTransportInfoEN.directions.**GoogleDirections**(*input_json={}*, ***kwargs*)
Bases: alex.applications.PublicTransportInfoEN.directions.Directions

Traffic directions obtained from Google Maps API.

class alex.applications.PublicTransportInfoEN.directions.**GoogleDirectionsFinder**(*cfg*)
Bases: alex.applications.PublicTransportInfoEN.directions.DirectionsFinder, alex.tools.apirequest.APIRequest

Transit direction finder using the Google Maps query engine.

get_directions(**args*, ***kwds*)
Get Google maps transit directions between the given stops at the given time and date.

The time/date should be given as a datetime.datetime object. Setting the correct date is compulsory!

map_vehicle(*vehicle*)
maps PTIEN vehicle type to GOOGLE DIRECTIONS query vehicle

class alex.applications.PublicTransportInfoEN.directions.**GoogleRoute**(*input_json*)
Bases: alex.applications.PublicTransportInfoEN.directions.Route

class alex.applications.PublicTransportInfoEN.directions.**GoogleRouteLeg**(*input_json*)
Bases: alex.applications.PublicTransportInfoEN.directions.RouteLeg

class alex.applications.PublicTransportInfoEN.directions.**GoogleRouteLegStep**(*input_json*)
Bases: alex.applications.PublicTransportInfoEN.directions.RouteStep

VEHICLE_TYPE_MAPPING = {u'FUNICULAR': u'cable_car', u'COMMUTER_TRAIN': u'train', u'INTERCITY_BUS

class alex.applications.PublicTransportInfoEN.directions.**Route**
Bases: object

Ancestor class for one transit direction route.

class alex.applications.PublicTransportInfoEN.directions.**RouteLeg**
Bases: object

One traffic directions leg.

class alex.applications.PublicTransportInfoEN.directions.**RouteStep**(*travel_mode*)
Bases: object

One transit directions step – walking or using public transport. Data members: travel_mode – TRANSIT / WALKING

- **For TRANSIT steps:** departure_stop departure_time arrival_stop arrival_time headsign – direction of the transit line vehicle – type of the transit vehicle (tram, subway, bus) line_name – name or number of the transit line
- **For WALKING steps:** duration – estimated walking duration (seconds)

MODE_TRANSIT = u'TRANSIT'

MODE_WALKING = u'WALKING'

class alex.applications.PublicTransportInfoEN.directions.**Travel**(***kwargs*)
Bases: object

Holder for starting and ending point (and other parameters) of travel.

get_minimal_info()
Return minimal waypoints information in the form of a stringified inform() dialogue act.

alex.applications.PublicTransportInfoEN.exceptions module

exception alex.applications.PublicTransportInfoEN.exceptions.**PTIENHDCPolicyException**
Bases: alex.components.dm.exceptions.DialoguePolicyException

alex.applications.PublicTransportInfoEN.hdc_policy module

alex.applications.PublicTransportInfoEN.hdc_slu module

alex.applications.PublicTransportInfoEN.preprocessing module

alex.applications.PublicTransportInfoEN.site_preprocessing module

alex.applications.PublicTransportInfoEN.site_preprocessing.**expand**(*element*, *spell_numbers=True*)

alex.applications.PublicTransportInfoEN.site_preprocessing.**expand_stop**(*stop*, *spell_numbers=True*)

alex.applications.PublicTransportInfoEN.site_preprocessing.**fix_ordinal**(*word*)

alex.applications.PublicTransportInfoEN.site_preprocessing.**spell_if_number**(*word*, *use_coupling*, *ordinal=True*)

alex.applications.PublicTransportInfoEN.test_hdc_policy module

alex.applications.PublicTransportInfoEN.test_hdc_slu module

alex.applications.PublicTransportInfoEN.time_zone module

class alex.applications.PublicTransportInfoEN.time_zone.**GoogleTimeFinder**(*cfg*)
Bases: alex.tools.apirequest.APIRequest

get_time(*place=None*, *lat=None*, *lon=None*)
Get time information at given place

obtain_geo_codes(*place=u'New York'*)
: :return: Returns tuple (longitude, latitude) for given place. Default value for place is New York

parse_time(*response*)

class alex.applications.PublicTransportInfoEN.time_zone.**Time**
Bases: object

Module contents

alex.applications.utils package

Submodules

alex.applications.utils.weather module

class alex.applications.utils.weather.**OpenWeatherMapWeather**(*input_json*, *condition_transl*, *date=None*, *daily=False*, *celsius=True*)
Bases: alex.applications.utils.weather.Weather

class alex.applications.utils.weather.**OpenWeatherMapWeatherFinder**(*cfg*)
Bases: alex.applications.utils.weather.WeatherFinder, alex.tools.apirequest.APIRequest

Weather service using OpenWeatherMap (http://openweathermap.org)

get_weather(**args*, ***kwds*)
Get OpenWeatherMap weather information or forecast for the given time.

The time/date should be given as a datetime.datetime object.

load(*file_name*)

class alex.applications.utils.weather.**Weather**
Bases: object

class alex.applications.utils.weather.**WeatherFinder**
Bases: object

Abstract ancestor for transit direction finders.

get_weather(*time=None*, *daily=False*, *place=None*)
Retrieve the weather for the given time, or for now (if time is None).

Should be implemented in derived classes.

class alex.applications.utils.weather.**WeatherPoint**(*in_city=None*, *in_state=None*)
Bases: object

Module contents

Submodules

alex.applications.ahub module

alex.applications.autopath module

self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.applications.exceptions module

exception `alex.applications.exceptions.`**`HubException`**
Bases: `alex.AlexException`

exception `alex.applications.exceptions.`**`SemHubException`**
Bases: `alex.applications.exceptions.HubException`

exception `alex.applications.exceptions.`**`TextHubException`**
Bases: `alex.applications.exceptions.HubException`

exception `alex.applications.exceptions.`**`VoipHubException`**
Bases: `alex.applications.exceptions.HubException`

alex.applications.shub module

alex.applications.thub module

alex.applications.vhub module

alex.applications.webhub module

Module contents

alex.components package

Subpackages

alex.components.asr package

Submodules

alex.components.asr.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.components.asr.base module

alex.components.asr.common module

alex.components.asr.common.**asr_factory**(*cfg*, *asr_type=None*)
Returns instance of specified ASR decoder in asr_type.

The ASR decoders are imported on the fly, because they need external non Python libraries.

alex.components.asr.common.**get_asr_type**(*cfg*)
Reads the ASR type from the configuration.

alex.components.asr.exceptions module

exception alex.components.asr.exceptions.**ASRException**
Bases: alex.AlexException

exception alex.components.asr.exceptions.**JuliusASRException**
Bases: alex.components.asr.exceptions.ASRException

exception alex.components.asr.exceptions.**JuliusASRTimeoutException**
Bases: alex.components.asr.exceptions.ASRException

exception alex.components.asr.exceptions.**KaldiASRException**
Bases: alex.components.asr.exceptions.ASRException

exception alex.components.asr.exceptions.**KaldiSetupException**
Bases: alex.components.asr.exceptions.KaldiASRException

alex.components.asr.google module

alex.components.asr.pykaldi module

alex.components.asr.test_utterance module

alex.components.asr.utterance module

Module contents

alex.components.dm package

Submodules

alex.components.dm.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.components.dm.base module

class alex.components.dm.base.**DialogueManager**(*cfg*)
Bases: object

This is a base class for a dialogue manager. The purpose of a dialogue manager is to accept input in the form dialogue acts and respond again in the form of dialogue acts.

The dialogue manager should be able to accept multiple inputs without producing any output and be able to produce multiple outputs without any input.

da_in(*da*, *utterance=None*)
Receives an input dialogue act or dialogue act list with probabilities or dialogue act confusion network.

When the dialogue act is received an update of the state is performed.

da_out()
Produces output dialogue act.

end_dialogue()
Ends the dialogue and post-process the data.

log_state()
Log the state of the dialogue state.

Returns none

new_dialogue()
Initialises the dialogue manager and makes it ready for a new dialogue conversation.

class alex.components.dm.base.**DialoguePolicy**(*cfg*, *ontology*)
Bases: object

This is a base class policy.

get_da(*dialogue_state*)

class alex.components.dm.base.**DialogueState**(*cfg*, *ontology*)
Bases: object

This is a trivial implementation of a dialogue state and its update.

It uses only the best dialogue act from the input and based on this it updates its state.

get_slots_being_confirmed()
Returns all slots which are currently being confirmed by the user along with the value being confirmed.

get_slots_being_noninformed()
Returns all slots provided by the user and the system has not informed about them yet along with the value of the slot.

get_slots_being_requested()
Returns all slots which are currently being requested by the user along with the correct value.

log_state()
Log the state using the the session logger.

restart()
Reinitialises the dialogue state so that the dialogue manager can start from scratch.

Nevertheless, remember the turn history.

update(*user_da*, *system_da*)
Interface for the dialogue act update.

It can process dialogue act, dialogue act N best lists, or dialogue act confusion networks.

Parameters

- **user_da** (DialogueAct, DialogueActNBList or DialogueActConfusionNetwork) – Dialogue act to process.
- **system_da** – Last system dialogue act.

class alex.components.dm.base.**DiscreteValue**(*values*, *name=''*, *desc=''*)
Bases: object

explain(*full=False*, *linear_prob=False*)
This function prints the values and their probabilities for this node.

mph()
The function returns the most probable value and its probability in a tuple.

mpv()
The function returns the most probable value.

mpvp()
The function returns the probability of the most probable value.

normalise()
This function normalise the sum of all probabilities to 1.0

prune(*threshold=0.001*)
Prune all values with probability less then a threshold.

tmphs()
This function returns two most probable values and their probabilities.

The function returns a tuple consisting of two tuples (probability, value).

tmpvs()
The function returns two most probable values.

tmpvsp()
The function returns probabilities of two most probable values in the slot.

alex.components.dm.common module
alex.components.dm.common.**dm_factory**(*dm_type*, *cfg*)
alex.components.dm.common.**get_dm_type**(*cfg*)

alex.components.dm.dddstate module

alex.components.dm.dstc_tracker module

alex.components.dm.dummypolicy module

alex.components.dm.exceptions module

exception alex.components.dm.exceptions.**DMException**
Bases: alex.AlexException

exception alex.components.dm.exceptions.**DeterministicDiscriminativeDialogueStateException**
Bases: alex.components.dm.exceptions.DialogueStateException

exception alex.components.dm.exceptions.**DialogueManagerException**
Bases: alex.AlexException

exception alex.components.dm.exceptions.**DialoguePolicyException**
Bases: alex.AlexException

exception alex.components.dm.exceptions.**DialogueStateException**
Bases: alex.AlexException

exception alex.components.dm.exceptions.**DummyDialoguePolicyException**
Bases: alex.components.dm.exceptions.DialoguePolicyException

alex.components.dm.ontology module

class alex.components.dm.ontology.**Ontology**(*file_name=None*)
Bases: object

Represents an ontology for a dialogue domain.

get_compatible_vals(*slot_pair*, *value*)
Given a slot pair (key to 'compatible_values' in ontology data), this returns the set of compatible values for the given key. If there is no information about the given pair, None is returned.

Parameters

- **slot_pair** – key to 'compatible_values' in ontology data
- **value** – the subkey to check compatible values for

Return type set

get_default_value(*slot*)
Given a slot name, get its default value (if set in the ontology). Returns None if the default value is not set for the given slot.

Parameters **slot** – the name of the desired slot

Return type unicode

is_compatible(*slot_pair*, *val1*, *val2*)
Given a slot pair and a pair of values, this tests whether the values are compatible. If there is no information about the slot pair or the first value, returns False. If the second value is None, returns always True (i.e. None is compatible with anything).

Parameters

- **slot_pair** – key to 'compatible_values' in ontology data
- **val1** – value of the 1st slot

- **val2** – value of the 2nd slot

Return type Boolean

last_talked_about(**args*, ***kwds*)
Returns a list of slots and values that should be used to for tracking about what was talked about recently, given the input dialogue acts.

Parameters

- **da_type** – the source dialogue act type
- **name** – the source slot name
- **value** – the source slot value

Returns returns a list of target slot names and values used for tracking

load(*file_name*)

reset_on_change(**args*, ***kwds*)

slot_has_value(*name*, *value*)
Check whether the slot and the value are compatible.

slot_is_binary(*name*)
Check whether the given slot has a binary value (using the 'binary' key in the 'slot_attributes' for the given slot name).

Parameters **name** – name of the slot being checked

slots_system_confirms(**args*, ***kwds*)
Return all slots the system can request.

slots_system_requests(**args*, ***kwds*)
Return all slots the system can request.

slots_system_selects(**args*, ***kwds*)
Return all slots the system can request.

exception alex.components.dm.ontology.**OntologyException**
Bases: exceptions.Exception

alex.components.dm.pstate module

class alex.components.dm.pstate.**PDDiscrete**(*initial=None*)
Bases: alex.components.dm.pstate.PDDiscreteBase

Discrete probability distribution.

NULL = None

OTHER = '<other>'

get(*item*)

get_distrib()

get_entropy()

get_items()

iteritems()

meta_slots = set([None, '<other>'])

normalize()
Normalize the probability distribution.

update(*items*)

class alex.components.dm.pstate.**PDDiscreteBase**(**args*, ***kwargs*)
Bases: object

get_best()

get_max(*which_one=0*)

remove(*item*)

class alex.components.dm.pstate.**PDDiscreteOther**(*space_size*, *initial=None*)
Bases: alex.components.dm.pstate.PDDiscreteBase

Discrete probability distribution with sink probability slot for OTHER.

NULL = None

OTHER = '<other>'

get(*item*)

get_distrib()

get_entropy()

get_items()

get_max(*which_one=0*)

iteritems()

meta_slots = set([None, '<other>'])

normalize(*redistrib=0.0*)
Normalize the probability distribution.

space_size = None

update(*items*)

class alex.components.dm.pstate.**SimpleUpdater**(*slots*)
Bases: object

update(*observ*)

update_slot(*slot*, *observ_distrib*)

alex.components.dm.state module

class alex.components.dm.state.**State**(*slots*)
Bases: object

update(*item*, *value*)

alex.components.dm.tracker module

class alex.components.dm.tracker.**StateTracker**
Bases: object

state_class = None

update_state(*state*, *cn*)
Update state according to the confusion network cn.

Module contents

alex.components.hub package

Submodules

alex.components.hub.ahub module

alex.components.hub.aio module

alex.components.hub.asr module

alex.components.hub.calldb module

class alex.components.hub.calldb.**CallDB**(*cfg*, *file_name*, *period=86400*)

Bases: object

Implements logging of all interesting call stats. It can be used for customization of the SDS, e.g. for novice or expert users.

close_database(*db*)

get_uri_stats(*remote_uri*)

log()

log_uri(*remote_uri*)

open_database()

read_database()

release_database()

track_confirmed_call(*remote_uri*)

track_disconnected_call(*remote_uri*)

alex.components.hub.dm module

alex.components.hub.exceptions module

exception alex.components.hub.exceptions.**VoipIOException**

Bases: alex.AlexException

alex.components.hub.hub module

class alex.components.hub.hub.**Hub**(*cfg*)

Bases: object

Common functionality for the hubs.

hub_type = 'Hub'

init_readline()

Initialize the readline functionality to enable console history.

write_readline()

alex.components.hub.messages module

class alex.components.hub.messages.**ASRHyp**(*hyp*, *source=None*, *target=None*, *fname=None*)
Bases: alex.components.hub.messages.Message

class alex.components.hub.messages.**Command**(*command*, *source=None*, *target=None*)
Bases: alex.components.hub.messages.Message

class alex.components.hub.messages.**DMDA**(*da*, *source=None*, *target=None*)
Bases: alex.components.hub.messages.Message

class alex.components.hub.messages.**Frame**(*payload*, *source=None*, *target=None*)
Bases: alex.components.hub.messages.Message

class alex.components.hub.messages.**Message**(*source*, *target*)
Bases: alex.utils.mproc.InstanceID

Abstract class which implements basic functionality for messages passed between components in the alex.

get_time_str()
Return current time in dashed ISO-like format.

class alex.components.hub.messages.**SLUHyp**(*hyp*, *asr_hyp=None*, *source=None*, *target=None*)
Bases: alex.components.hub.messages.Message

class alex.components.hub.messages.**TTSText**(*text*, *source=None*, *target=None*)
Bases: alex.components.hub.messages.Message

alex.components.hub.nlg module

alex.components.hub.slu module

alex.components.hub.tts module

alex.components.hub.vad module

alex.components.hub.vio module

alex.components.hub.webio module

Module contents

alex.components.nlg package

Subpackages

alex.components.nlg.tectotpl package

Subpackages

alex.components.nlg.tectotpl.block package

Subpackages

alex.components.nlg.tectotpl.block.a2w package

Subpackages

alex.components.nlg.tectotpl.block.a2w.cs package

Submodules

alex.components.nlg.tectotpl.block.a2w.cs.concatenatetokens module

class alex.components.nlg.tectotpl.block.a2w.cs.concatenatetokens.**ConcatenateTokens**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

Detokenize the sentence, spread whitespace correctly.

process_zone(*zone*)

Detokenize the sentence and assign the result to the sentence attribute of the current zone.

alex.components.nlg.tectotpl.block.a2w.cs.removerepeatedtokens module

class alex.components.nlg.tectotpl.block.a2w.cs.removerepeatedtokens.**RemoveRepeatedTokens**(*scen*
arg

Bases: alex.components.nlg.tectotpl.core.block.Block

Remove two identical neighboring tokens.

process_zone(*zone*)

Remove two identical neighboring tokens in the given sentence.

Module contents

Module contents

alex.components.nlg.tectotpl.block.read package

Submodules

alex.components.nlg.tectotpl.block.read.tectotemplates module

class alex.components.nlg.tectotpl.block.read.tectotemplates.**TectoTemplates**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

Reader for partial t-tree dialog system templates, where treelets can be intermixed with linear text.

Example template:

Vlak přijede v [[7|adj:attr] hodina|n:4|gender:fem].

All linear text is inserted into t-lemmas of atomic nodes, while treelets have their formeme and grammateme values filled in.

parse_line(*text*, *troot*)
Parse a template to a t-tree.

parse_treelet(*text*, *tnode*)
Parse a treelet in the template, filling the required values. Returns the position in the text after the treelet.

process_document(*filename*)
Read a Tecto-Template file and return its contents as a Document object.

alex.components.nlg.tectotpl.block.read.yaml module

class alex.components.nlg.tectotpl.block.read.yaml.**YAML**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

process_document(*filename*)
Read a YAML file and return its contents as a Document object

Module contents

alex.components.nlg.tectotpl.block.t2a package

Subpackages

alex.components.nlg.tectotpl.block.t2a.cs package

Submodules

alex.components.nlg.tectotpl.block.t2a.cs.addappositionpunct module

class alex.components.nlg.tectotpl.block.t2a.cs.addappositionpunct.**AddAppositionPunct**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

Separating Czech appositions, such as in 'John, my best friend, ...' with commas.

Arguments: language: the language of the target tree selector: the selector of the target tree

add_comma_node(*aparent*)
Add a comma a-node to the given parent

is_before_punct(*anode*)
Test whether the subtree of the given node precedes a punctuation node.

process_tnode(*tnode*)
Adds punctuation a-nodes if the given node is an apposition node.

alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundfuture module

class alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundfuture.**AddAuxVerbCompoundFu**

Bases: alex.components.nlg.tectotpl.core.block.Block

Add compound future auxiliary 'bude'.

Arguments: language: the language of the target tree selector: the selector of the target tree

process_tnode(*tnode*)
Add compound future auxiliary to a node, where appropriate.

alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpassive module

class alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpassive.**AddAuxVerbCompoundP**

Bases: alex.components.nlg.tectotpl.core.block.Block

Add compound passive auxiliary 'být'.

Arguments: language: the language of the target tree selector: the selector of the target tree

process_tnode(*tnode*)

Add compound passive auxiliary to a node, where appropriate.

alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpast module

class alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpast.**AddAuxVerbCompoundPast**

Bases: alex.components.nlg.tectotpl.core.block.Block

Add compound past tense auxiliary of the 1st and 2nd person 'jsem/jsi/jsme/jste'.

Arguments: language: the language of the target tree selector: the selector of the target tree

AUX_PAST_FORMS = {(u'P', u'2'): u'jste', (u'S', u'1'): u'jsem', (u'S', u'2'): u'jsi', (u'.', u'2'): u'jsi', (u'P', u'1'): u'jsme'

process_tnode(*tnode*)

Add compound past auxiliary to a node, where appropriate.

alex.components.nlg.tectotpl.block.t2a.cs.addauxverbconditional module

class alex.components.nlg.tectotpl.block.t2a.cs.addauxverbconditional.**AddAuxVerbConditional**(*s*
a

Bases: alex.components.nlg.tectotpl.core.block.Block

Add conditional auxiliary 'by'/'bych'.

Arguments: language: the language of the target tree selector: the selector of the target tree

process_tnode(*tnode*)

Add conditional auxiliary to a node, where appropriate.

alex.components.nlg.tectotpl.block.t2a.cs.addauxverbmodal module

class alex.components.nlg.tectotpl.block.t2a.cs.addauxverbmodal.**AddAuxVerbModal**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

Add modal verbs.

Arguments: language: the language of the target tree selector: the selector of the target tree

DEONTMOD_2_MODAL = {u'vol': u'cht\xedt', u'hrt': u'm\xedt', u'perm': u'moci', u'fac': u'moci', u'deb': u'muset', u'po

process_tnode(*tnode*)

Add modal auxiliary to a node, where appropriate.

alex.components.nlg.tectotpl.block.t2a.cs.addclausalexpletives module

class alex.components.nlg.tectotpl.block.t2a.cs.addclausalexpletives.**AddClausalExpletives**(*scer*
args

Bases: alex.components.nlg.tectotpl.block.t2a.addauxwords.AddAuxWords

Add clausal expletive pronoun 'to' (+preposition) to subordinate clauses with 'že', if the parent verb requires it.

Arguments: language: the language of the target tree selector: the selector of the target tree

get_anode(*tnode*)
Return the a-node that is the root of the verbal a-subtree.

get_aux_forms(*tnode*)
Return the clausal expletive to be added, if supposed to.

new_aux_node(*anode*, *form*)
Create a node for the expletive/its preposition.

postprocess(*tnode*, *anode*, *aux_anodes*)
Rehang the conjunction 'že', now above the expletive, under it. Fix clause numbers and ordering.

alex.components.nlg.tectotpl.block.t2a.cs.addclausalpunct module

class alex.components.nlg.tectotpl.block.t2a.cs.addclausalpunct.**AddClausalPunct**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

An abstract ancestor for blocks working with clausal punctuation.

Arguments: language: the language of the target tree selector: the selector of the target tree

is_clause_in_quotes(*anode*)
Return True if the given node is in an enquoted clause. The node must end the clause.

alex.components.nlg.tectotpl.block.t2a.cs.addcoordpunct module

class alex.components.nlg.tectotpl.block.t2a.cs.addcoordpunct.**AddCoordPunct**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

Add comma to coordinated lists of 3 and more elements, as well as before some Czech coordination conjunctions ('ale', 'ani').

Arguments: language: the language of the target tree selector: the selector of the target tree

add_comma_node(*anode*)
Add a comma AuxX node under the given node.

is_at_clause_boundary(*anode*)
Return true if the given node is at a clause boundary (i.e. the nodes immediately before and after it belong to different clauses).

process_anode(*anode*)
Add coordination punctuation to the given anode, if applicable.

alex.components.nlg.tectotpl.block.t2a.cs.addparentheses module

class alex.components.nlg.tectotpl.block.t2a.cs.addparentheses.**AddParentheses**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

Add '(' / ')' nodes to nodes which have the wild/is_parenthesis attribute set.

Arguments: language: the language of the target tree selector: the selector of the target tree

add_parenthesis_node(*anode*, *lemma*, *clause_num*)
Add a parenthesis node as a child of the specified a-node; with the given lemma and clause number set.

continued_paren_left(*anode*)
Return True if this node is continuing a parenthesis from the left.

continued_paren_right(*anode*)
Return True if a parenthesis continues after this node to the right.

process_anode(*anode*)
Add parentheses to an a-node, where appropriate.

alex.components.nlg.tectotpl.block.t2a.cs.addprepositions module

class alex.components.nlg.tectotpl.block.t2a.cs.addprepositions.**AddPrepositions**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.addauxwords.AddAuxWords

Add prepositional a-nodes according to formemes.

Arguments: language: the language of the target tree selector: the selector of the target tree

get_aux_forms(*tnode*)
Find prepositional nodes to be created.

new_aux_node(*anode*, *form*)
Create a prepositional node with the given preposition form and parent.

postprocess(*tnode*, *anode*, *aux_nodes*)
Move rhematizers in front of the newly created PPs.

alex.components.nlg.tectotpl.block.t2a.cs.addreflexiveparticles module

class alex.components.nlg.tectotpl.block.t2a.cs.addreflexiveparticles.**AddReflexiveParticles**(*s*
a

Bases: alex.components.nlg.tectotpl.core.block.Block

Add reflexive particles to reflexiva tantum and reflexive passive verbs.

Arguments: language: the language of the target tree selector: the selector of the target tree

process_tnode(*tnode*)
Add reflexive particle to a node, if applicable.

alex.components.nlg.tectotpl.block.t2a.cs.addsentfinalpunct module

class alex.components.nlg.tectotpl.block.t2a.cs.addsentfinalpunct.**AddSentFinalPunct**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.cs.addclausalpunct.AddClausalPunct

Add final sentence punctuation ('?', '.').

Arguments: language: the language of the target tree selector: the selector of the target tree

process_ttree(*troot*)
Add final punctuation to the given sentence.

alex.components.nlg.tectotpl.block.t2a.cs.addsubconjs module

class alex.components.nlg.tectotpl.block.t2a.cs.addsubconjs.**AddSubconjs**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.addauxwords.AddAuxWords

Add subordinate conjunction a-nodes according to formemes.

Arguments: language: the language of the target tree selector: the selector of the target tree

get_aux_forms(*tnode*)
Find prepositional nodes to be created.

new_aux_node(*anode*, *form*)
Create a subordinate conjunction node with the given conjunction form and parent.

alex.components.nlg.tectotpl.block.t2a.cs.addsubordclausepunct module

class alex.components.nlg.tectotpl.block.t2a.cs.addsubordclausepunct.**AddSubordClausePunct**(*scen*
args

Bases: alex.components.nlg.tectotpl.block.t2a.cs.addclausalpunct.AddClausalPunct

Add commas separating subordinate clauses.

Arguments: language: the language of the target tree selector: the selector of the target tree

are_in_coord_clauses(*aleft*, *aright*)
Check if the given nodes are in two coordinated clauses.

get_clause_parent(*anode*)
Return the parent of the clause the given node belongs to; the result may be the root of the tree.

insert_comma_between(*aleft*, *aright*)
Insert a comma node between these two nodes, find out where to hang it.

process_atree(*aroot*)
Add subordinate clause punctuation to the given sentence.

alex.components.nlg.tectotpl.block.t2a.cs.capitalizesentstart module

class alex.components.nlg.tectotpl.block.t2a.cs.capitalizesentstart.**CapitalizeSentStart**(*scenar*
args)

Bases: alex.components.nlg.tectotpl.core.block.Block

Capitalize the first word in the sentence (skip punctuation etc.).

OPEN_PUNCT = u'^[({[\u201a\u201e\xab\u2039|*"\\\']+$'

process_zone(*zone*)
Find the first valid word in the sentence and capitalize it.

alex.components.nlg.tectotpl.block.t2a.cs.deletesuperfluousauxs module

class alex.components.nlg.tectotpl.block.t2a.cs.deletesuperfluousauxs.**DeleteSuperfluousAuxs**(*s*
a

Bases: alex.components.nlg.tectotpl.core.block.Block

Delete repeated prepositions and and conjunctions in coordinations.

BASE_DIST_LIMIT = 8

DIST_LIMIT = {u'mezi': 50, u'pro': 8, u'proto\u017ee': 5, u'v': 5}

process_tnode(*tnode*)
Check for repeated prepositions and and conjunctions in coordinations, delete them if necessary.

alex.components.nlg.tectotpl.block.t2a.cs.dropsubjpersprons module

class alex.components.nlg.tectotpl.block.t2a.cs.dropsubjpersprons.**DropSubjPersProns**(*scenario*,
args)

Bases: alex.components.nlg.tectotpl.core.block.Block

Remove the Czech pro-drop subject personal pronouns (or demonstrative "to") from the a-tree.

Arguments: language: the language of the target tree selector: the selector of the target tree

drop_anode(*tnode*)
Remove the lexical a-node corresponding to the given t-node

process_tnode(*tnode*)
Check if the a-node corresponding to the given t-node should be dropped, and do so where appropriate.

alex.components.nlg.tectotpl.block.t2a.cs.generatepossessiveadjectives module

class alex.components.nlg.tectotpl.block.t2a.cs.generatepossessiveadjectives.**GeneratePossessi**

Bases: alex.components.nlg.tectotpl.core.block.Block

According to formemes, this changes the lemma of the surface possessive adjectives from the original (deep) lemma which was identical to the noun from which the adjective is derived, e.g. changes the a-node lemma from 'Čapek' to 'Čapkův' if the corresponding t-node has the 'adj:poss' formeme.

Arguments: language: the language of the target tree selector: the selector of the target tree

load()

process_tnode(*tnode*)
Check a t-node if its lexical a-node should be changed; if yes, update its lemma.

alex.components.nlg.tectotpl.block.t2a.cs.generatewordforms module

alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr module

class alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr.**ImposeAttrAgr**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.imposeagreement.ImposeAgreement

Impose case, gender and number agreement of attributes with their governing nouns.

Arguments: language: the language of the target tree selector: the selector of the target tree

impose(*tnode*, *match_nodes*)
Impose case, gender and number agreement on attributes.

process_excepts(*tnode*, *match_nodes*)
Handle special cases for this rule: nic/něco, numerals.

should_agree(*tnode*)
Find adjectives with a noun parent. Returns the a-layer nodes for the adjective and its parent, or False

alex.components.nlg.tectotpl.block.t2a.cs.imposecomplagr module

class alex.components.nlg.tectotpl.block.t2a.cs.imposecomplagr.**ImposeComplAgr**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.imposeagreement.ImposeAgreement

Impose agreement of adjectival verb complements with the subject.

Arguments: language: the language of the target tree selector: the selector of the target tree

impose(*tnode*, *match_nodes*)
Impose the agreement on selected adjectival complements.

process_excepts(*tnode*, *match_nodes*)
Returns False; there are no special cases for this rule.

should_agree(*tnode*)
Find the complement and its subject.

alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr module

class alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr.**ImposePronZAgr**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.imposeagreement.ImposeAgreement

In phrases such as 'každý z ...','žádná z ...', impose agreement in gender.

Arguments: language: the language of the target tree selector: the selector of the target tree

PRONOUNS = u'^(jeden|ka\u017ed\xfd|\u017e\xe1dn\xfd|oba|v\u0161echen|(n\u011b|lec)kter\xfd|(jak|kter)\xfdkoliv?|libov

impose(*tnode*, *tchild*)
Impose the gender agreement on selected nodes.

process_excepts(*tnode*, *match_nodes*)
Returns False; there are no special cases for this rule.

should_agree(*tnode*)
Find matching pronouns with 'z+2'-formeme children.

alex.components.nlg.tectotpl.block.t2a.cs.imposerelpronagr module

class alex.components.nlg.tectotpl.block.t2a.cs.imposerelpronagr.**ImposeRelPronAgr**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.imposeagreement.ImposeAgreement

Impose gender and number agreement of relative pronouns with their antecedent.

Arguments: language: the language of the target tree selector: the selector of the target tree

impose(*tnode*, *tantec*)
Impose the gender agreement on selected nodes.

process_excepts(*tnode*, *match_nodes*)
Returns False; there are no special cases for this rule.

should_agree(*tnode*)
Find relative pronouns with a valid antecedent.

alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpredagr module

class alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpredagr.**ImposeSubjPredAgr**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.block.t2a.imposeagreement.ImposeAgreement

Impose gender and number agreement of relative pronouns with their antecedent.

Arguments: language: the language of the target tree selector: the selector of the target tree

impose(*tnode*, *match_nodes*)
Impose the subject-predicate agreement on regular nodes.

process_excepts(*tnode*, *match_nodes*)
Returns False; there are no special cases for this rule.

should_agree(*tnode*)
Find finite verbs, with/without a subject.

alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat module

class alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.**InitMorphcat**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

According to t-layer grammatemes, this initializes the morphcat structure at the a-layer that is the basis for a later POS tag limiting in the word form generation.

Arguments: language: the language of the target tree selector: the selector of the target tree

DEGREE = {u'comp': u'2', u'pos': u'1', u'acomp': u'2', u'sup': u'3', None: u'.', u'nr': u'.'}

GENDER = {u'anim': u'M', u'fem': u'F', u'inan': u'I', u'inher': u'.', u'neut': u'N', None: u'.', u'nr': u'.'}

NEGATION = {None: u'A', u'neg0': u'A', u'neg1': u'N'}

NUMBER = {None: u'.', u'nr': u'.', u'sg': u'S', u'pl': u'P', u'inher': u'.'}

PERSON = {u'1': u'1', None: u'.', u'3': u'3', u'2': u'2', u'inher': u'.'}

VOICE = {u'pas': u'P', u'deagent': u'A', u'passive': u'P', u'act': u'A', u'active': u'A', None: u'.'}

process_tnode(*tnode*)
Initialize the morphcat structure in the given node

set_case(*tnode*, *anode*)
Set the morphological case for an a-node according to the corresponding t-node's formeme, where applicable.

set_perspron_categories(*tnode*, *anode*)
Set detailed morphological categories of personal pronouns of various types (possessive, reflexive, personal per se)

alex.components.nlg.tectotpl.block.t2a.cs.marksubject module

class alex.components.nlg.tectotpl.block.t2a.cs.marksubject.**MarkSubject**(*scenario*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

Marks the subject of each clause with the Afun 'Sb'.

Arguments: language: the language of the target tree selector: the selector of the target tree

process_ttree(*ttree*)
Mark all subjects in a sentence

alex.components.nlg.tectotpl.block.t2a.cs.markverbalcategories module

class alex.components.nlg.tectotpl.block.t2a.cs.markverbalcategories.**MarkVerbalCategories**(*scen*, *args*)

Bases: alex.components.nlg.tectotpl.core.block.Block

Finishes marking synthetic verbal categories: tense, finiteness, mood.

Arguments: language: the language of the target tree selector: the selector of the target tree

mark_subpos_tense(*tnode*, *anode*)
Marks the Sub-POS and tense parts of the morphcat structure in plain verbal a-nodes.

process_tnode(*tnode*)
Marks verbal categories for a t-node.

resolve_imperative(*anode*)
Mark an imperative a-node.

resolve_infinitive(*anode*)
Mark an infinitive a-node correctly.

alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowackernagel module

class alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowackernagel.**MoveCliticsToWackerr**

Bases: alex.components.nlg.tectotpl.core.block.Block

Move clitics (e.g. 'se', 'to' etc.) to the second (Wackernagel) position in the clause.

clitic_order(*clitic*)
Return the position of the given clitic in the natural Czech order of multiple clitics in the same clause.

find_eo1st_pos(*clause_root*, *clause_1st*)
Find the last word before the Wackernagel position.

handle_pronoun_je(*anode*)
If the given node is a personal pronoun with the form 'je', move it before its parent's subtree and return True. Return false otherwise.

is_clitic(*anode*)
Return True if the given node belongs to a clitic.

is_coord_taking_1st_pos(*clause_root*)
Return True if the clause root is a coordination member and the coordinating conjunction or shared subjunction is taking up the 1st position. E.g. 'Běžel, aby se zahřál a dostal se dřív domů.'

process_atree(*aroot*)
Process the individual clauses – find and move clitics within them.

process_clause(*clause*)
Find and move clitics within one clause.

should_ignore(*anode*, *clause_number*)
Return True if this word should be ignored in establishing the Wackernagel position.

verb_group_root(*clitic*)
Find the root of the verbal group that the given clitic belongs to. If the verbal group is governed by a conjunction, return this conjunction.

alex.components.nlg.tectotpl.block.t2a.cs.projectclausenumber module

class alex.components.nlg.tectotpl.block.t2a.cs.projectclausenumber.**ProjectClauseNumber**(*scenar*
args)

Bases: alex.components.nlg.tectotpl.core.block.Block

Project clause numbering from t-nodes to a-nodes.

Arguments: language: the language of the target tree selector: the selector of the target tree

process_tnode(*tnode*)
Project the t-node's clause number to all its corresponding a-nodes.

alex.components.nlg.tectotpl.block.t2a.cs.reversenumbernoundependency module

class alex.components.nlg.tectotpl.block.t2a.cs.reversenumbernoundependency.**ReverseNumberNour**

Bases: alex.components.nlg.tectotpl.core.block.Block

This block reverses the dependency of incongruent Czech numerals (5 and higher), hanging their parents under them in the a-tree.

Arguments: language: the language of the target tree selector: the selector of the target tree

process_ttree(*ttree*)
Rehang the numerals for the given t-tree & a-tree pair

alex.components.nlg.tectotpl.block.t2a.cs.vocalizeprepos module

class alex.components.nlg.tectotpl.block.t2a.cs.vocalizeprepos.**VocalizePrepos**(*scenario*,
args)

Bases: alex.components.nlg.tectotpl.core.block.Block

This block replaces the forms of prepositions 'k', 'v', 'z', 's' with their vocalized variants 'ke'/'ku', 've', 'ze', 'se' according to the following word.

process_atree(*aroot*)
Find and vocalize prepositions according to their context.

vocalize(*prep*, *follow*)
Given a preposition lemma and the form of the word following it, return the appropriate form (base or vocalized).

Module contents

Submodules

alex.components.nlg.tectotpl.block.t2a.addauxwords module

class alex.components.nlg.tectotpl.block.t2a.addauxwords.**AddAuxWords**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

Add auxiliary a-nodes according to formemes.

This is a base class for all steps adding auxiliary nodes.

Arguments: language: the language of the target tree selector: the selector of the target tree

get_anode(*tnode*)
Return the a-node corresponding to the given t-node. Defaults to lexical a-node.

get_aux_forms(*tnode*)
This should return a list of new forms for the auxiliaries, or None if none should be added

new_aux_node(*aparent*, *form*)
Create an auxiliary node with the given surface form and parent.

postprocess(*tnode*, *anode*, *aux_nodes*)
Apply content-specific post-processing to the newly created auxiliary a-nodes (to be overridden if needed).

process_tnode(*tnode*)
Add auxiliary words to the a-layer for a t-node.

alex.components.nlg.tectotpl.block.t2a.copyttree module

class alex.components.nlg.tectotpl.block.t2a.copyttree.**CopyTTree**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

This block creates an a-tree based on a t-tree in the same zone.

Arguments: language: the language of the target zone selector: the selector of the target zone

copy_subtree(*troot*, *aroot*)
Deep-copy a subtree, creating nodes with the same attributes, but different IDs.

process_zone(*zone*)
Starting tree copy

alex.components.nlg.tectotpl.block.t2a.imposeagreement module

class alex.components.nlg.tectotpl.block.t2a.imposeagreement.**ImposeAgreement**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

A common ancestor for blocks that impose a grammatical agreement of some kind: they should override the should_agree(tnode), process_excepts(tnode), and impose(tnode) methods.

Arguments: language: the language of the target tree selector: the selector of the target tree

impose(*tnode*, *match_nodes*)
Impose the agreement onto the given (regular) node.

process_excepts(*tnode*, *match_nodes*)
Process exceptions from the agreement. If an exception has been found and impose() should not fire, return True.

process_tnode(*tnode*)
Impose the required agreement on a node, if applicable.

should_agree(*tnode*)
Check whether the agreement applies to the given node; if so, return the relevant nodes this node should agree with.

Module contents

alex.components.nlg.tectotpl.block.t2t package

Module contents

alex.components.nlg.tectotpl.block.util package

Submodules

alex.components.nlg.tectotpl.block.util.copytree module

class alex.components.nlg.tectotpl.block.util.copytree.**CopyTree**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

This block is able to copy a tree on the same layer from a different zone.

Arguments: language: the language of the TARGET zone selector: the selector of the TARGET zone source_language the language of the SOURCE zone (defaults to same as target) source_selector the selector of the SOURCE zone (defaults to same as target) layer: the layer to which this conversion should be applied

TODO: apply to more layers at once

copy_subtree(*source_root*, *target_root*)
Deep-copy a subtree, creating nodes with the same attributes, but different IDs

process_bundle(*bundle*)
For each bundle, copy the tree on the given layer in the given zone to another zone.

alex.components.nlg.tectotpl.block.util.eval module

class alex.components.nlg.tectotpl.block.util.eval.**Eval**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

This block executes arbitrary Python code for each document/bundle or each zone/tree/node matching the current language and selector.

Arguments:

document, bundle, zone, atree, anode, ttree, tnode, ntree, nnode, ptree, pnode: code to execute for each <name of the argument>

Arguments may be combined, but at least one of them must be set. If only X<tree/node> are set, language and selector is required.

process_bundle(*bundle*)
Process a document (execute code from the 'bundle' argument and dive deeper)

process_document(*doc*)
Process a document (execute code from the 'document' argument and dive deeper)

process_zone(*zone*)
Process a zone (according to language and selector; execute code for the zone or X<tree|node>) arguments)

valid_args = [u'document', u'doc', u'bundle', u'zone', u'atree', u'anode', u'ttree', u'tnode', u'ntree', u'nnode', u'ptre

alex.components.nlg.tectotpl.block.util.setglobal module

class alex.components.nlg.tectotpl.block.util.setglobal.**SetGlobal**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

process_bundle(*doc*)
This block does nothing with the documents, its only work is setting the global arguments in the initialization phase.

Module contents

alex.components.nlg.tectotpl.block.write package

Submodules

alex.components.nlg.tectotpl.block.write.basewriter module

class alex.components.nlg.tectotpl.block.write.basewriter.**BaseWriter**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.core.block.Block

Base block for output writing.

get_output_file_name(*doc*)
Create an output file name for the given document.

alex.components.nlg.tectotpl.block.write.yaml module

class alex.components.nlg.tectotpl.block.write.yaml.**YAML**(*scenario*, *args*)
Bases: alex.components.nlg.tectotpl.block.write.basewriter.BaseWriter

default_extension = u'.yaml'

process_document(*doc*)
Write a YAML document

serialize_bundle(*bundle*)
Serialize a bundle to a list.

serialize_node(*node*, *add_parent_id*)
Serialize a node to a hash; using the correct attributes for the tree type given. Add the node parent's id if needed.

serialize_tree(*root*)

serialize_zone(*zone*)
Serialize a zone into a hash

Module contents

Module contents

alex.components.nlg.tectotpl.core package

Submodules

alex.components.nlg.tectotpl.core.block module

class alex.components.nlg.tectotpl.core.block.**Block**(*scenario*, *args*)
Bases: object

A common ancestor to all Treex processing blocks.

load()
Load required files / models, to be overridden by child blocks.

process_bundle(*bundle*)
Process a bundle. Default behavior is to process the zone according to the current language and selector.

process_document(*doc*)
Process a document. Default behavior is to look for methods that process a bundle/zone/tree/node. If none is found, raise a NotImplementedError.

process_zone(*zone*)
Process a zone. Default behavior is to try if there is a process_Xtree or process_Xnode method and run this method, otherwise raise an error.

alex.components.nlg.tectotpl.core.document module

class alex.components.nlg.tectotpl.core.document.**Bundle**(*document*, *data=None*, *b_ord=None*)
Bases: object

Represents a bundle, i.e. a list of zones pertaining to the same sentence (in different variations).

create_zone(*language*, *selector*)
Creates a zone at the given language and selector. Will overwrite any existing zones.

document
The document this bundle belongs to.

get_all_zones()
Return all zones contained in this bundle.

get_or_create_zone(*language*, *selector*)
Returns the zone for a language and selector; if it does not exist, creates an empty zone.

get_zone(*language*, *selector*)
Returns the corresponding zone for a language and selector; raises an exception if the zone does not exist.

has_zone(*language*, *selector*)
Returns True if the bundle has a zone for the given language and selector.

ord
The order of this bundle in the document, as given by constructor

class alex.components.nlg.tectotpl.core.document.**Document**(*filename=None*, *data=None*)

Bases: object

This represents a Treex document, i.e. a sequence of bundles. It contains an index of node IDs.

create_bundle(*data=None*)
Append a new bundle and return it.

get_node_by_id(*node_id*)

index_backref(*attr_name*, *source_id*, *target_ids*)
Keep track of a backward reference (source, target node IDs are in the direction of the original reference)

index_node(*node*)
Index a node by its id. Also index the node's references in the backwards reference index.

remove_backref(*attr_name*, *source_id*, *target_ids*)
Remove references from the backwards index.

remove_node(*node_id*)
Remove a node from all indexes.

class alex.components.nlg.tectotpl.core.document.**Zone**(*data=None*, *language=None*, *selector=None*, *bundle=None*)

Bases: object

Represents a zone, i.e. a sentence and corresponding trees.

atree
Direct access to a-tree (will raise an exception if the tree does not exist).

bundle
The bundle in which this zone is located

create_atree()
Create a tree on the a-layer

create_ntree()
Create a tree on the n-layer

create_ptree()
Create a tree on the p-layer

create_tree(*layer*, *data=None*)
Create a tree on the given layer, filling it with the given data (if applicable).

create_ttree()
Create a tree on the t-layer

document
The document in which this zone is located

get_tree(*layer*)
Return a tree this node has on the given layer or raise an exception if the tree does not exist.

has_atree()
Return true if this zone has an a-tree.

has_ntree()
Return true if this zone has an n-tree.

has_ptree()
Return true if this zone has a p-tree.

has_tree(*layer*)
Return True if this zone has a tree on the given layer, False otherwise.

has_ttree()
Return true if this zone has a t-tree.

language_and_selector
Return string concatenation of the zone's language and selector.

ntree
Direct access to n-tree (will raise an exception if the tree does not exist).

ptree
Direct access to p-tree (will raise an exception if the tree does not exist).

ttree
Direct access to t-tree (will raise an exception if the tree does not exist).

alex.components.nlg.tectotpl.core.exception module

exception alex.components.nlg.tectotpl.core.exception.**DataException**(*path*)
Bases: alex.components.nlg.tectotpl.core.exception.TreexException

Data file not found exception

exception alex.components.nlg.tectotpl.core.exception.**LoadingException**(*text*)
Bases: alex.components.nlg.tectotpl.core.exception.TreexException

Block loading exception

exception alex.components.nlg.tectotpl.core.exception.**RuntimeException**(*text*)
Bases: alex.components.nlg.tectotpl.core.exception.TreexException

Block runtime exception

exception alex.components.nlg.tectotpl.core.exception.**ScenarioException**(*text*)
Bases: alex.components.nlg.tectotpl.core.exception.TreexException

Scenario-related exception.

exception alex.components.nlg.tectotpl.core.exception.**TreexException**(*message*)
Bases: exceptions.Exception

Common ancestor for Treex exception

alex.components.nlg.tectotpl.core.log module

alex.components.nlg.tectotpl.core.log.**log_info**(*message*)
Print an information message

alex.components.nlg.tectotpl.core.log.**log_warn**(*message*)
Print a warning message

alex.components.nlg.tectotpl.core.node module

class alex.components.nlg.tectotpl.core.node.**A**(*data=None*, *parent=None*, *zone=None*)
Bases: alex.components.nlg.tectotpl.core.node.Node, alex.components.nlg.tectotpl.core.node
alex.components.nlg.tectotpl.core.node.EffectiveRelations,
alex.components.nlg.tectotpl.core.node.InClause

Representing an a-node

attrib = [(u'form', <type 'unicode'>), (u'lemma', <type 'unicode'>), (u'tag', <type 'unicode'>), (u'afun', <type 'unicode

is_coap_root()

morphcat_case

morphcat_gender

morphcat_grade

morphcat_members = [u'pos', u'subpos', u'gender', u'number', u'case', u'person', u'tense', u'negation', u'voice', u'gr

morphcat_mood

morphcat_negation

morphcat_number

morphcat_person

morphcat_pos

morphcat_possgender

morphcat_possnumber

morphcat_subpos

morphcat_tense

morphcat_voice

ref_attrib = [u'p_terminal.rf']

reset_morphcat()
Reset the morphcat structure members to '.'

class alex.components.nlg.tectotpl.core.node.**EffectiveRelations**
Bases: object

Representing a node with effective relations

attrib = [(u'is_member', <type 'bool'>)]

get_coap_members()
Return the members of the coordination, if the node is a coap root. Otherwise return the node itself.

get_echildren(*or_topological=False*, *add_self=False*, *ordered=False*, *preceding_only=False*, *following_only=False*)
Return the effective children of the current node.

get_eparents(*or_topological=False*, *add_self=False*, *ordered=False*, *preceding_only=False*, *following_only=False*)
Return the effective parents of the current node.

is_coap_root()
Testing whether the node is a coordination/apposition root. Must be implemented in descendants.

ref_attrib = []

class alex.components.nlg.tectotpl.core.node.**InClause**
Bases: object

Represents nodes that are organized in clauses

attrib = [(u'clause_number', <type 'int'>), (u'is_clause_head', <type 'bool'>)]

get_clause_root()
Return the root of the clause the current node resides in.

ref_attrib = []

class alex.components.nlg.tectotpl.core.node.**N**(*data=None*, *parent=None*, *zone=None*)
Bases: alex.components.nlg.tectotpl.core.node.Node

Representing an n-node

attrib = [(u'ne_type', <type 'unicode'>), (u'normalized_name', <type 'unicode'>), (u'a.rf', <type 'list'>)]

ref_attrib = [u'a.rf']

class alex.components.nlg.tectotpl.core.node.**Node**(*data=None*, *parent=None*, *zone=None*)
Bases: object

Representing a node in a tree (recursively)

attrib = [(u'alignment', <type 'list'>), (u'wild', <type 'dict'>)]

create_child(*id=None*, *data=None*)
Create a child of the current node

document
The document this node is a member of.

get_attr(*name*)
Return the value of the given attribute. Allows for dictionary nesting, e.g. 'morphcat/gender'

get_attr_list(*include_types=False*, *safe=False*)
Get attributes of the current class (gathering all attributes of base classes)

get_children(*add_self=False*, *ordered=False*, *preceding_only=False*, *following_only=False*)
Return all children of the node

get_depth()
Return the depth, i.e. the distance to the root.

get_deref_attr(*name*)
This assumes the given attribute holds node id(s) and returns the corresponding node(s)

get_descendants(*add_self=False*, *ordered=False*, *preceding_only=False*, *following_only=False*)
Return all topological descendants of this node.

get_ref_attr_list(*split_nested=False*)
Return a list of the attributes of the current class that contain references (splitting nested ones, if needed)

get_referenced_ids()
Return all ids referenced by this node, keyed under their reference types in a hash.

id
The unique id of the node within the document.

is_root
Return true if this node is a root

parent
The parent of the current node. None for roots.

ref_attrib = []

remove()
Remove the node from the tree.

remove_reference(*ref_type*, *refd_id*)
Remove the reference of the given type to the given node.

root
The root of the tree this node is in.

set_attr(*name*, *value*)
Set the value of the given attribute. Allows for dictionary nesting, e.g. 'morphcat/gender'

set_deref_attr(*name*, *value*)
This assumes the value is a node/list of nodes and sets its id/their ids as the value of the given attribute.

zone
The zone this node belongs to.

class alex.components.nlg.tectotpl.core.node.**Ordered**
Bases: object

Representing an ordered node (has an attribute called ord), defines sorting.

attrib = [(u'ord', <type 'int'>)]

get_next_node()
Get the following node in the ordering.

get_prev_node()
Get the preceding node in the ordering.

is_first_node()
Return True if this node is the first node in the tree, i.e. has no previous nodes.

is_last_node()
Return True if this node is the last node in the tree, i.e. has no following nodes.

is_right_child
Return True if this node has a greater ord than its parent. Returns None for a root.

ref_attrib = []

shift_after_node(*other*, *without_children=False*)
Shift one node after another in the ordering.

shift_after_subtree(*other*, *without_children=False*)
Shift one node after the whole subtree of another node in the ordering.

shift_before_node(*other*, *without_children=False*)
Shift one node before another in the ordering.

shift_before_subtree(*other*, *without_children=False*)
Shift one node before the whole subtree of another node in the ordering.

class alex.components.nlg.tectotpl.core.node.**P**(*data=None*, *parent=None*, *zone=None*)
Bases: alex.components.nlg.tectotpl.core.node.Node

Representing a p-node

attrib = [(u'is_head', <type 'bool'>), (u'index', <type 'unicode'>), (u'coindex', <type 'unicode'>), (u'edgelabel', <type '

ref_attrib = []

class alex.components.nlg.tectotpl.core.node.**T**(*data=None*, *parent=None*, *zone=None*)
Bases: alex.components.nlg.tectotpl.core.node.Node, alex.components.nlg.tectotpl.core.node
alex.components.nlg.tectotpl.core.node.EffectiveRelations,
alex.components.nlg.tectotpl.core.node.InClause

Representing a t-node

add_aux_anodes(*new_anodes*)
Add an auxiliary a-node/a-nodes to the list.

anodes
Return all anodes of a t-node

attrib = [(u'functor', <type 'unicode'>), (u'formeme', <type 'unicode'>), (u't_lemma', <type 'unicode'>), (u'nodetype',

aux_anodes

compl_nodes

coref_gram_nodes

coref_text_nodes

gram_aspect

gram_degcmp

gram_deontmod

gram_diathesis

gram_dispmod

gram_gender

gram_indeftype

gram_iterativeness

gram_negation

gram_number

gram_numertype

gram_person

gram_politeness

gram_resultative

gram_sempos

gram_tense

gram_verbmod

is_coap_root()

lex_anode

ref_attrib = [u'a/lex.rf', u'a/aux.rf', u'compl.rf', u'coref_gram.rf', u'coref_text.rf']

remove_aux_anodes(*to_remove*)
Remove an auxiliary a-node from the list

alex.components.nlg.tectotpl.core.run module

class alex.components.nlg.tectotpl.core.run.**Scenario**(*config*)
Bases: object

This represents a scenario, i.e. a sequence of blocks to be run on the data

apply_to(*string*, *language=None*, *selector=None*)
Apply the whole scenario to a string (which should be readable by the first block of the scenario), return the sentence(s) of the given target language and selector.

load_blocks()
Load all blocks into memory, finding and creating class objects.

alex.components.nlg.tectotpl.core.util module

alex.components.nlg.tectotpl.core.util.**as_list**(*value*)
Cast anything to a list (just copy a list or a tuple, or put an atomic item to as a single element to a list).

alex.components.nlg.tectotpl.core.util.**file_stream**(*filename*, *mode=u'r'*, *encoding=u'UTF-8'*)
Given a file stream or a file name, return the corresponding stream, handling GZip. Depending on mode, open an input or output stream.

alex.components.nlg.tectotpl.core.util.**first**(*condition_function*, *sequence*, *default=None*)
Return first item in sequence where condition_function(item) == True, or None if no such item exists.

Module contents

alex.components.nlg.tectotpl.tool package

Subpackages

alex.components.nlg.tectotpl.tool.lexicon package

Submodules

alex.components.nlg.tectotpl.tool.lexicon.cs module

class alex.components.nlg.tectotpl.tool.lexicon.cs.**Lexicon**
Bases: object

get_possessive_adj_for(*noun_lemma*)
Given a noun lemma, this returns a possessive adjective if it's in the database.

has_expletive(*lemma*)
Return an expletive for a 'že'-clause that this verb governs, or False. Lemmas must include reflexive particles for reflexiva tantum.

has_synthetic_future(*verb_lemma*)
Returns True if the verb builds a synthetic future tense form with the prefix 'po-'/'pů-'.

inflect_conditional(*lemma*, *number*, *person*)
Return inflected form of a conditional particle/conjunction

is_coord_conj(*lemma*)
Return 'Y'/'N' if the given lemma is a coordinating conjunction (depending on whether one should write a comma directly in front).

is_incongruent_numeral(*numeral*)
Return True if the given lemma belongs to a Czech numeral that takes a genitive attribute instead of being an attribute itself

is_named_entity_label(*lemma*)
Return 'I'/'C' if the given lemma is a named entity label (used as congruent/incongruent attribute).

is_personal_role(*lemma*)
Return true if the given lemma is a personal role.

load_possessive_adj_dict(*data_dir*)
Read the possessive-adjective-to-noun conversion file and save it to the database.

number_for(*numeral*)
Given a Czech numeral, returns the corresponding number.

Module contents

alex.components.nlg.tectotpl.tool.ml package

Submodules

alex.components.nlg.tectotpl.tool.ml.dataset module

alex.components.nlg.tectotpl.tool.ml.model module

Module contents

Submodules

alex.components.nlg.tectotpl.tool.cluster module

class alex.components.nlg.tectotpl.tool.cluster.**Job**(*code=None*, *header=u'#!/usr/bin/env pythonn# coding=utf8nfrom __future__ import unicode_literalsn'*, *name=None*, *work_dir=None*, *dependencies=None*)

Bases: object

This represents a piece of code as a job on the cluster, holds information about the job and is able to retrieve job metadata.

The most important method is submit(), which submits the given piece of code to the cluster.

Important attributes (some may be set in the constructor or at job submission, but all may be set between construction and launch): ———————————————————— name – job name on the cluster (and the name of the created

Python script, default will be generated if not set)

code – the Python code to be run (needs to have imports and sys.path set properly)

header – the header of the created Python script (may contain imports etc.)

memory – the amount of memory to reserve for this job on the cluster

cores – the number of cores needed for this job work_dir – the working directory where the job script will be

created and run (will be created on launch)

dependencies-list of Jobs this job depends on (must be submitted before submitting this job)

In addition, the following values may be queried for each job at runtime or later: ———————————————————— submitted – True if the job has been submitted to the cluster. state – current job state ('qw' = queued, 'r' = running, 'f'

= finished, only if the job was submitted)

host – the machine where the job is running (short name) jobid – the numeric id of the job in the cluster (NB: type is

string!)

report – job report using the qacct command (dictionary, available only after the job has finished)

exit_status- numeric job exit status (if the job is finished)

DEFAULT_CORES = 1

DEFAULT_HEADER = u'#!/usr/bin/env python\n# coding=utf8\nfrom __future__ import unicode_literals\n'

DEFAULT_MEMORY = 4

DIR_PREFIX = u'_clrun-'

FINISH = u'f'

JOBNAME_LEGAL_CHARS = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'

NAME_PREFIX = u'pyjob_'

QSUB_MEMORY_CMD = u'-hard -l mem_free={0} -l act_mem_free={0} -l h_vmem={0}'

QSUB_MULTICORE_CMD = u'-pe smp {0}'

TIME_POLL_DELAY = 60

TIME_QUERY_DELAY = 1

add_dependency(*dependency*)
Adds a dependency on the given Job(s).

exit_status
Retrieve the exit status of the job via the qacct report. Throws an exception the job is still running and the exit status is not known.

get_script_text()
Join headers and code to create a meaningful Python script.

host
Retrieve information about the host this job is/was running on.

jobid
Return the job id.

name
Return the job name.

remove_dependency(*dependency*)
Removes the given Job(s) from the dependencies list.

report
Access to qacct report. Please note that running the qacct command takes a few seconds, so the first access to the report is rather slow.

state
Retrieve information about current job state. Will also retrieve the host this job is running on and store it in the __host variable, if applicable.

submit (*memory=None*, *cores=None*, *work_dir=None*)
Submit the job to the cluster. Override the pre-set memory and cores defaults if necessary. The job code, header and working directory must be set in advance. All jobs on which this job is dependent must already be submitted!

wait (*poll_delay=None*)
Waits for the job to finish. Will raise an exception if the job did not finish successfully. The poll_delay variable controls how often the job state is checked.

Module contents

Module contents

alex.components.nlg.tools package

Submodules

alex.components.nlg.tools.cs module

alex.components.nlg.tools.en module A collection of helper functions for generating English.

alex.components.nlg.tools.en.**every_word_for_number**(*number*, *ordinary=False*, *use_coupling=False*)

params: ordinary - if set to True, it returns ordinal of the number (fifth rather than five etc).

use_coupling if set to True, it returns number greater than 100 with "and" between hundreds and tens (two hundred and seventeen rather than two hundred seventeen).

Returns a word given a number 1-100

alex.components.nlg.tools.en.**word_for_number**(*number*, *ordinary=False*)
Returns a word given a number 1-100

Module contents

Submodules

alex.components.nlg.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.components.nlg.common module

alex.components.nlg.exceptions module

exception alex.components.nlg.exceptions.**NLGException**
Bases: alex.AlexException

exception alex.components.nlg.exceptions.**TemplateNLGException**
Bases: alex.components.nlg.exceptions.NLGException

alex.components.nlg.template module

alex.components.nlg.test_tectotpl module

alex.components.nlg.test_template module

Module contents

alex.components.slu package

Submodules

alex.components.slu.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.components.slu.base module

alex.components.slu.common module

alex.components.slu.cued_da module

alex.components.slu.da module

alex.components.slu.dailrclassifier module

alex.components.slu.dainnclassifier module

alex.components.slu.exceptions module

exception alex.components.slu.exceptions.**CuedDialogueActError**
Bases: alex.components.slu.exceptions.SLUException

exception alex.components.slu.exceptions.**DAIKernelException**
Bases: alex.components.slu.exceptions.SLUException

exception alex.components.slu.exceptions.**DAILRException**
Bases: alex.components.slu.exceptions.SLUException

exception alex.components.slu.exceptions.**DialogueActConfusionNetworkException**
Bases: alex.components.slu.exceptions.SLUException, alex.ml.hypothesis.ConfusionNetworkExc

exception alex.components.slu.exceptions.**DialogueActException**
Bases: alex.components.slu.exceptions.SLUException

exception alex.components.slu.exceptions.**DialogueActItemException**
Bases: alex.components.slu.exceptions.SLUException

exception alex.components.slu.exceptions.**DialogueActNBListException**
Bases: alex.components.slu.exceptions.SLUException

exception alex.components.slu.exceptions.**SLUConfigurationException**
Bases: alex.components.slu.exceptions.SLUException

exception alex.components.slu.exceptions.**SLUException**
Bases: alex.AlexException

alex.components.slu.templateclassifier module

class alex.components.slu.templateclassifier.**TemplateClassifier**(*config*)
Bases: object

This parser is based on matching examples of utterances with known semantics against input utterance. The semantics of the example utterance which is closest to the input utterance is provided as a output semantics.

"Hi" => hello() "I can you give me a phone number" => request(phone) "I would like to have a phone number please" => request(phone)

The first match is reported as the resulting dialogue act.

parse(*asr_hyp*)

readRules(*file_name*)

alex.components.slu.test_da module

alex.components.slu.test_dailrclassifier module

alex.components.slu.test_dainnclassifier module

Module contents

alex.components.tts package

Submodules

alex.components.tts.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.components.tts.base module

class alex.components.tts.base.**TTSInterface**(*cfg*)

Bases: object

synthesize(*text*)

alex.components.tts.common module

alex.components.tts.exceptions module

exception alex.components.tts.exceptions.**TTSException**

Bases: alex.AlexException

alex.components.tts.flite module

alex.components.tts.google module

alex.components.tts.preprocessing module

class alex.components.tts.preprocessing.**TTSPreprocessing**(*cfg*, *file_name*)

Bases: object

Preprocess words that are hard to pronounce for the current TTS engine.

load(*file_name*)

process(*text*)

Applies all substitutions on the input text and returns the result.

class alex.components.tts.preprocessing.**TTSPreprocessingException**

Bases: object

alex.components.tts.speechtech module

alex.components.tts.test_google module

alex.components.tts.test_voicerss module

alex.components.tts.voicerss module

Module contents

alex.components.vad package

Submodules

alex.components.vad.ffnn module

alex.components.vad.gmm module

alex.components.vad.power module

class alex.components.vad.power.**PowerVAD**(*cfg*)

This is implementation of a simple power based voice activity detector.

It only implements simple decisions whether input frame is speech of non speech.

decide(*frame*)

Returns whether the input segment is speech or non speech.

The returned values can be in range from 0.0 to 1.0. It returns 1.0 for 100% speech segment and 0.0 for 100% non speech segment.

Module contents

Module contents

alex.corpustools package

Submodules

alex.corpustools.asr_decode module

alex.corpustools.asrscore module

alex.corpustools.autopath module

self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.corpustools.cued-audio2ufal-audio module

alex.corpustools.cued-call-logs-sem2ufal-call-logs-sem module

alex.corpustools.cued-sem2ufal-sem module

alex.corpustools.cued module

This module is meant to collect functionality for handling call logs – both working with the call log files in the filesystem, and parsing them.

alex.corpustools.cued.**find_logs**(*infname*, *ignore_list_file=None*, *verbose=False*)

Finds CUED logs below the paths specified and returns their filenames. The logs are determined as files matching one of the following patterns:

user-transcription.norm.xml user-transcription.xml user-transcription-all.xml

If multiple patterns are matched by files in the same directory, only the first match is taken.

Arguments:

infname – either a directory, or a file. In the first case, logs are looked for below that directory. In the latter case, the file is read line by line, each line specifying a directory or a glob determining the log to include.

ignore_list_file – a file of absolute paths or globs (can be mixed) specifying logs that should be excluded from the results

verbose – print lots of output?

Returns a set of paths to files satisfying the criteria.

alex.corpustools.cued.**find_wavs**(*infname*, *ignore_list_file=None*)

Finds wavs below the paths specified and returns their filenames.

Arguments:

infname – either a directory, or a file. In the first case, wavs are looked for below that directory. In the latter case, the file is read line by line, each line specifying a directory or a glob determining the wav to include.

ignore_list_file – a file of absolute paths or globs (can be mixed) specifying wavs that should be excluded from the results

Returns a set of paths to files satisfying the criteria.

alex.corpustools.cued.**find_with_ignorelist**(*infname*, *pat*, *ignore_list_file=None*, *find_kwargs={}*)

Finds specific files below the paths specified and returns their filenames.

Arguments: pat – globbing pattern specifying the files to look for infname – either a directory, or a file. In the first case, wavs are

looked for below that directory. In the latter case, the file is read line by line, each line specifying a directory or a glob determining the wav to include.

ignore_list_file – a file of absolute paths or globs (can be mixed) specifying wavs that should be excluded from the results

find_kwargs – if provided, this dictionary is used as additional keyword arguments for the function 'utils.fs.find' for finding positive examples of files (not the ignored ones)

Returns a set of paths to files satisfying the criteria.

alex.corpustools.cued2utt_da_pairs module

class alex.corpustools.cued2utt_da_pairs.**TurnRecord**
Bases: tuple

TurnRecord(transcription, cued_da, cued_dahyp, asrhyp, audio)

asrhyp
Alias for field number 3

audio
Alias for field number 4

cued_da
Alias for field number 1

cued_dahyp
Alias for field number 2

transcription
Alias for field number 0

alex.corpustools.cued2utt_da_pairs.**extract_trns_sems**(*infname*, *verbose*, *fields=None*, *ignore_list_file=None*, *do_exclude=True*, *normalise=True*, *known_words=None*)
Extracts transcriptions and their semantic annotation from a directory containing CUED call log files.

Arguments:

infname – either a directory, or a file. In the first case, logs are looked for below that directory. In the latter case, the file is read line by line, each line specifying a directory or a glob determining the call log to include.

verbose – print lots of output? fields – names of fields that should be required for the output.

Field names are strings corresponding to the element names in the transcription XML format. (default: all five of them)

ignore_list_file – a file of absolute paths or globs (can be mixed) specifying logs that should be skipped

normalise – whether to do normalisation on transcriptions do_exclude – whether to exclude transcriptions not considered suitable known_words – a collection of words. If provided, transcriptions are

excluded which contain other words. If not provided, excluded are transcriptions that contain any of _excluded_characters. What "excluded" means depends on whether the transcriptions are required by being specified in 'fields'.

Returns a list of TurnRecords.

alex.corpustools.cued2utt_da_pairs.**extract_trns_sems_from_file**(*fname*, *verbose*, *fields=None*, *normalise=True*, *do_exclude=True*, *known_words=None*, *robust=False*)

Extracts transcriptions and their semantic annotation from a CUED call log file.

Arguments: fname – path towards the call log file verbose – print lots of output? fields – names of fields that should be required for the output.

Field names are strings corresponding to the element names in the transcription XML format. (default: all five of them)

normalise – whether to do normalisation on transcriptions do_exclude – whether to exclude transcriptions not considered suitable known_words – a collection of words. If provided, transcriptions are

excluded which contain other words. If not provided, excluded are transcriptions that contain any of _excluded_characters. What "excluded" means depends on whether the transcriptions are required by being specified in 'fields'.

robust – whether to assign recordings to turns robustly or trust where they are in the log. This could be useful for older CUED logs where the elements sometimes escape to another <turn> than they belong. However, in cases where 'robust' leads to finding the correct recording for the user turn, the log is damaged at other places too, and the resulting turn record would be misleading. Therefore, we recommend leaving robust=False.

Returns a list of TurnRecords.

alex.corpustools.cued2utt_da_pairs.**write_asrhyp_sem**(*outdir*, *fname*, *data*)

alex.corpustools.cued2utt_da_pairs.**write_asrhyp_semhyp**(*outdir*, *fname*, *data*)

alex.corpustools.cued2utt_da_pairs.**write_data**(*outdir*, *fname*, *data*, *tpt*)

alex.corpustools.cued2utt_da_pairs.**write_trns_sem**(*outdir*, *fname*, *data*)

alex.corpustools.cued2wavaskey module

Finds CUED XML files describing calls in the directory specified, extracts a couple of fields from them for each turn (transcription, ASR 1-best, semantics transcription, SLU 1-best) and outputs them to separate files in the following format:

{wav_filename} => {field}

An example ignore list file could contain the following three lines:

/some-path/call-logs/log_dir/some_id.wav some_id.wav jurcic-??[13579]*.wav

The first one is an example of an ignored path. On UNIX, it has to start with a slash. On other platforms, an analogic convention has to be used.

The second one is an example of a literal glob.

The last one is an example of a more advanced glob. It says basically that all odd dialogue turns should be ignored.

alex.corpustools.cued2wavaskey.**main**(*args*)

alex.corpustools.cuedda module

class alex.corpustools.cuedda.**CUEDDialogueAct**(*text*, *da*, *database=None*, *dictionary=None*)

get_cued_da()

get_slots_and_values()

get_ufal_da()

parse()

class alex.corpustools.cuedda.**CUEDSlot**(*slot*)

parse()

alex.corpustools.fisherptwo2ufal-audio module

alex.corpustools.grammar_weighted module

class alex.corpustools.grammar_weighted.**A**(**rules*)
Bases: alex.corpustools.grammar_weighted.Alternative

class alex.corpustools.grammar_weighted.**Alternative**(**rules*)
Bases: alex.corpustools.grammar_weighted.Rule

sample()

class alex.corpustools.grammar_weighted.**GrammarGen**(*root*)
Bases: object

sample(*n*)
Sampling of n sentences.

sample_uniq(*n*)
Unique sampling of n sentences.

class alex.corpustools.grammar_weighted.**O**(*rule*, *prob=0.5*)
Bases: alex.corpustools.grammar_weighted.Option

class alex.corpustools.grammar_weighted.**Option**(*rule*, *prob=0.5*)
Bases: alex.corpustools.grammar_weighted.Rule

sample()

class alex.corpustools.grammar_weighted.**Rule**
Bases: object

class alex.corpustools.grammar_weighted.**S**(**rules*)
Bases: alex.corpustools.grammar_weighted.Sequence

class alex.corpustools.grammar_weighted.**Sequence**(**rules*)
Bases: alex.corpustools.grammar_weighted.Rule

sample()

class alex.corpustools.grammar_weighted.**T**(*string*)
Bases: alex.corpustools.grammar_weighted.Terminal

class alex.corpustools.grammar_weighted.**Terminal**(*string*)
Bases: alex.corpustools.grammar_weighted.Rule

sample()

class alex.corpustools.grammar_weighted.**UA**(**rules*)
Bases: alex.corpustools.grammar_weighted.UniformAlternative

class alex.corpustools.grammar_weighted.**UniformAlternative**(**rules*)
Bases: alex.corpustools.grammar_weighted.Rule

load(*fn*)
Load alternative terminal strings from a file.

Parameters **fn** – a file name

sample()

alex.corpustools.grammar_weighted.**as_terminal**(*rule*)

alex.corpustools.grammar_weighted.**as_weight_tuple**(*rule*, *def_weight=1.0*)

alex.corpustools.grammar_weighted.**clamp_01**(*number*)

alex.corpustools.grammar_weighted.**counter_weight**(*rules*)

alex.corpustools.grammar_weighted.**remove_spaces**(*utterance*)

alex.corpustools.librispeech2ufal-audio module

alex.corpustools.lm module

alex.corpustools.malach-en2ufal-audio module

alex.corpustools.merge_uttcns module

alex.corpustools.merge_uttcns.**find_best_cn**(*cns*)
Determines which one of decoded confnets seems the best.

alex.corpustools.merge_uttcns.**merge_files**(*fnames*, *outfname*)

alex.corpustools.num_time_stats module

Traverses the filesystem below a specified directory, looking for call log directories. Writes a file containing statistics about each phone number (extracted from the call log dirs' names):

- number of calls
- total size of recorded wav files
- last expected date the caller would call
- last date the caller actually called
- the phone number

Call with -h to obtain the help for command line arguments.

2012-12-11 Matěj Korvas

alex.corpustools.num_time_stats.**get_call_data_from_fs**(*rootdir*)

alex.corpustools.num_time_stats.**get_call_data_from_log**(*log_fname*)

alex.corpustools.num_time_stats.**get_timestamp**(*date*)
Total seconds in the timedelta.

alex.corpustools.num_time_stats.**mean**(*collection*)

alex.corpustools.num_time_stats.**sd**(*collection*)

alex.corpustools.num_time_stats.**set_and_ret**(*indexable*, *idx*, *val*)

alex.corpustools.num_time_stats.**var**(*collection*)

alex.corpustools.recording_splitter module

alex.corpustools.semscore module

alex.corpustools.semscore.**load_semantics**(*file_name*)

alex.corpustools.semscore.**score**(*fn_refsem*, *fn_testsem*, *item_level=False*, *detailed_error_output=False*, *outfile=<open file '<stdout>', mode 'w' at 0x7ff397c0e150>*)

alex.corpustools.semscore.**score_da**(*ref_da*, *test_da*, *daid*)
Computed according to http://en.wikipedia.org/wiki/Precision_and_recall

alex.corpustools.semscore.**score_file**(*refsem*, *testsem*)

alex.corpustools.split-asr-data module

alex.corpustools.srilm_ppl_filter module

alex.corpustools.srilm_ppl_filter.**main**()

alex.corpustools.srilm_ppl_filter.**srilm_scores**(*d3*)

alex.corpustools.text_norm_cs module

This module provides tools for **CZECH** normalisation of transcriptions, mainly for those obtained from human transcribers.

alex.corpustools.text_norm_cs.**normalise_text**(*text*)
Normalises the transcription. This is the main function of this module.

alex.corpustools.text_norm_cs.**exclude_by_dict**(*text*, *known_words*)
Determines whether text is not good enough and should be excluded.

"Good enough" is defined as having all its words present in the 'known_words' collection.

alex.corpustools.text_norm_en module

This module provides tools for **ENGLISH** normalisation of transcriptions, mainly for those obtained from human transcribers.

alex.corpustools.text_norm_en.**normalise_text**(*text*)
Normalises the transcription. This is the main function of this module.

alex.corpustools.text_norm_en.**exclude_by_dict**(*text*, *known_words*)
Determines whether text is not good enough and should be excluded.

"Good enough" is defined as having all its words present in the 'known_words' collection.

alex.corpustools.text_norm_es module

This module provides tools for **ENGLISH** normalisation of transcriptions, mainly for those obtained from human transcribers.

alex.corpustools.text_norm_es.**normalise_text**(*text*)
Normalises the transcription. This is the main function of this module.

alex.corpustools.text_norm_es.**exclude_by_dict**(*text*, *known_words*)
Determines whether text is not good enough and should be excluded.

"Good enough" is defined as having all its words present in the 'known_words' collection.

alex.corpustools.ufal-call-logs-audio2ufal-audio module

alex.corpustools.ufal-transcriber2ufal-audio module

alex.corpustools.ufaldatabase module

alex.corpustools.ufaldatabase.**save_database**(*odir*, *slots*)

alex.corpustools.vad-mlf-from-ufal-audio module

alex.corpustools.voxforge2ufal-audio module

alex.corpustools.wavaskey module

alex.corpustools.wavaskey.**load_wavaskey**(*fname*, *constructor*, *limit=None*, *encoding=u'UTF-8'*)
Loads a dictionary of objects stored in the "wav as key" format.

The input file is assumed to contain lines of the following form:

[[:space:]..]<key>[[:space:]..]=>[[:space:]..]<obj_str>[[:space:]..]

or just (without keys):

[[:space:]..]<obj_str>[[:space:]..]

where <obj_str> is to be given as the only argument to the 'constructor' when constructing the objects stored in the file.

Arguments: fname – path towards the file to read the objects from constructor – function that will be called on each string stored in

the file and whose result will become a value of the returned dictionary

limit – limit on the number of objects to read encoding – the file encoding

Returns a dictionary with objects constructed by 'constructor' as values.

alex.corpustools.wavaskey.**save_wavaskey**(*fname*, *in_dict*, *encoding=u'UTF-8'*, *trans=<function <lambda> at 0x7ff392ca47d0>*)

Saves a dictionary of objects in the wave as key format into a file.

Parameters

- **file_name** – name of the target file
- **utt** – a dictionary with the objects where the keys are the names of teh corresponding wave files

Parma trans a function which can transform a saved object

Returns None

Module contents

alex.ml package

Subpackages

alex.ml.bn package

Submodules

alex.ml.bn.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.ml.bn.factor module

alex.ml.bn.lbp module Belief propagation algorithms for factor graph.

class alex.ml.bn.lbp.**BP**
Bases: object

Abstract class for Belief Propagation algorithm.

run()
Run inference algorithm.

exception alex.ml.bn.lbp.**BPError**
Bases: exceptions.Exception

class alex.ml.bn.lbp.**LBP**(*strategy='sequential'*, ***kwargs*)
Bases: alex.ml.bn.lbp.BP

Loopy Belief Propagation.

LBP is an approximative inference algorithm for factor graphs. LBP works with generic factor graphs. It does accurate inference for trees and is equal to sum-product algorithm there.

It is possible to specify which strategy should be used for choosing next node for update. Sequential strategy will update nodes in exact order in which they were added. Tree strategy will assume the graph is a tree (without checking) and will do one pass of sum-product algorithm.

add_layer(*layer*)

add_layers(*layers*)
Add layers of nodes to graph.

add_nodes(*nodes*)
Add nodes to graph.

clear_layers()

clear_nodes()

init_messages()

run(*n_iterations=1*, *from_layer=None*)
Run the lbp algorithm.

exception alex.ml.bn.lbp.**LBPError**
Bases: alex.ml.bn.lbp.BPError

alex.ml.bn.node module

alex.ml.bn.test_factor module

alex.ml.bn.test_lbp module

alex.ml.bn.test_node module

alex.ml.bn.utils module

Module contents

alex.ml.ep package

Submodules

alex.ml.ep.node module

class alex.ml.ep.node.**ConstChangeGoal**(*name*, *desc*, *card*, *parameters*, *parents=None*)
Bases: alex.ml.ep.node.GroupingGoal

ConstChangeGoal implements all functionality as is include in GroupingGoal; however, it that there are only two transition probabilites for transitions between the same values and the different values.

update()
This function update belief for the goal.

class alex.ml.ep.node.**Goal**(*name*, *desc*, *card*, *parameters*, *parents=None*)
Bases: alex.ml.ep.node.Node

Goal can contain only the same values as the observations.

As a consequence, it can contain values of its previous node.

probTable(*value*, *parents*)
This function defines how the coditional probability is computed.

pRemebering - probability that the previous value is correct pObserving - probability that the observed value is correct

setParents(*parents*)

setValues()
The function copy values from its previous node and from observation nodes.

update()
This function update belief for the goal.

class alex.ml.ep.node.**GroupingGoal**(*name*, *desc*, *card*, *parameters*, *parents=None*)
Bases: alex.ml.ep.node.GroupingNode, alex.ml.ep.node.Goal

GroupingGoal implements all functionality as is include in Goal; however, it only update the values for which was observed some evidence.

setValues()
The function copy values from its previous node and from observation nodes.

update()
This function update belief for the goal.

class alex.ml.ep.node.**GroupingNode**(*name*, *desc*, *card*)
Bases: alex.ml.ep.node.Node

addOthers(*value*, *probability*)

explain(*full=None*)
This function explains the values for this node.

In additon to the Node's function, it prints the cardinality of the others set.

splitOff(*value*)
This function split off the value from the others set and place it into the values dict.

class alex.ml.ep.node.**Node**(*name*, *desc*, *card*)
Bases: object

A base class for all nodes in a belief state.

explain(*full=None*)
This function prints the values and their probailities for this node.

getMostProbableValue()
The function returns the most probable value and its probability in a tuple.

getTwoMostProbableValues()
This function returns two most probable values and their probabilities.

The function returns a tuple consisting of two tuples (value, probability).

normalise()
This function normlize the sum of all probabilities to 1.0

alex.ml.ep.test module
alex.ml.ep.test.**random**() → x in the interval [0, 1).

alex.ml.ep.turn module
class alex.ml.ep.turn.**Turn**

Module contents

alex.ml.gmm package

Submodules

alex.ml.gmm.gmm module

Module contents

alex.ml.lbp package

Submodules

alex.ml.lbp.node module

Module contents

Submodules

alex.ml.exceptions module

exception alex.ml.exceptions.**FFNNException**
Bases: alex.AlexException

exception alex.ml.exceptions.**NBListException**
Bases: alex.AlexException

alex.ml.features module

alex.ml.ffnn module

alex.ml.hypothesis module

This module collects classes representing the uncertainty about the actual value of a base type instance.

class alex.ml.hypothesis.**ConfusionNetwork**
Bases: alex.ml.hypothesis.Hypothesis

Confusion network. In this representation, each fact breaks down into a sequence of elementary acts.

add(*probability*, *fact*)
Append a fact to the confusion network.

add_merge(*p*, *fact*, *combine=u'max'*)
Add a fact and if it exists merge it according to the given combine strategy.

extend(*conf_net*)

classmethod from_fact(*fact*)
Constructs a deterministic confusion network that asserts the given 'fact'. Note that 'fact' has to be an iterable of elementary acts.

get_prob(*fact*)
Get the probability of the fact.

merge(*conf_net*, *combine=u'max'*)
Merges facts in the current and the given confusion networks.

Arguments:

combine – can be one of {'new', 'max', 'add', 'arit', 'harm'}, and determines how two probabilities should be merged (default: 'max')

XXX As of now, we know that different values for the same slot are contradictory (and in general, the set of contradicting attr-value pairs could be larger). We should therefore consider them alternatives to each other.

normalise()
Makes sure that all probabilities add up to one. They should implicitly sum to one: p + (1-p) == 1.0

prune(*prune_prob=0.005*)
Prune all low probability dialogue act items.

remove(*fact_to_remove*)

sort(*reverse=True*)

update_prob(*probability*, *fact*)
Update the probability of a fact.

exception alex.ml.hypothesis.**ConfusionNetworkException**
Bases: exceptions.Exception

class alex.ml.hypothesis.**Hypothesis**
Bases: object

This is the base class for all forms of probabilistic hypotheses representations.

classmethod from_fact(*fact*)
Constructs a deterministic hypothesis that asserts the given 'fact'.

class alex.ml.hypothesis.**NBList**
Bases: alex.ml.hypothesis.Hypothesis

This class represents the uncertainty using an n-best list.

When updating an N-best list, one should do the following.

1.add utterances or parse a confusion network

2.merge and normalise, in either order

add(*probability*, *fact*)
Finds the last hypothesis with a lower probability and inserts the new item before that one. Optimised for adding objects from the highest probability ones to the lowest probability ones.

add_other(*other*)
The N-best list is extended to include the other object to represent those object values that are not enumerated in the list.

Returns self.

classmethod from_fact(*fact*)

get_best()
Returns the most probable value of the object.

merge()
Adds up probabilities for the same hypotheses. Returns self.

normalise()
Scales the list to sum to one.

alex.ml.logarithmetic module

alex.ml.test_hypothesis module

class alex.ml.test_hypothesis.**TestConfusionNetwork**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_iter()

test_remove()

alex.ml.tffnn module

Module contents

alex.tests package

Submodules

alex.tests.autopath module

self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.tests.test_asr_google module

alex.tests.test_mproc module

alex.tests.test_numpy_with_optimised_ATLAS module

alex.tests.test_numpy_with_optimised_ATLAS.**main**()

alex.tests.test_pyaudio module

alex.tests.test_tts_flite_en module

alex.tests.test_tts_google_cs module

alex.tests.test_tts_google_en module

alex.tests.test_tts_voice_rss_en module

Module contents

alex.tools package

Subpackages

alex.tools.mturk package

Subpackages

alex.tools.mturk.bin package

Submodules

alex.tools.mturk.bin.approve_all_HITs module

alex.tools.mturk.bin.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.tools.mturk.bin.delete_all_HITs module

alex.tools.mturk.bin.delete_aproved_rejected_expired_HITs module

alex.tools.mturk.bin.expire_all_HITs module

alex.tools.mturk.bin.get_account_balance module

alex.tools.mturk.bin.mturk module

alex.tools.mturk.bin.mturk.**print_assignment**(*ass*)

alex.tools.mturk.bin.reject_HITs_from_worker module

Module contents

alex.tools.mturk.sds-evaluation package

Subpackages

alex.tools.mturk.sds-evaluation.common package

Submodules

alex.tools.mturk.sds-evaluation.common.lock_test module

alex.tools.mturk.sds-evaluation.common.mturk-ganalytics module

alex.tools.mturk.sds-evaluation.common.mturk-log module

alex.tools.mturk.sds-evaluation.common.mturk-logs-stats module

alex.tools.mturk.sds-evaluation.common.mturk-remote-addr module

alex.tools.mturk.sds-evaluation.common.utils module

Module contents

Submodules

alex.tools.mturk.sds-evaluation.autopath module

alex.tools.mturk.sds-evaluation.copy_feedbacks module

alex.tools.mturk.sds-evaluation.cued_feedback_stats module

alex.tools.mturk.sds-evaluation.cued_phone_number_stats module

Module contents

Module contents

alex.tools.vad package

Submodules

alex.tools.vad.autopath module self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

    import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

    pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.tools.vad.train_vad_gmm module

alex.tools.vad.train_vad_nn_theano module

Module contents

Submodules

alex.tools.apirequest module

class alex.tools.apirequest.**APIRequest**(*cfg*, *fname_prefix*, *log_elem_name*)
Bases: object

Handles functions related web API requests (logging).

class alex.tools.apirequest.**DummyLogger**(*stream=<open file '<stderr>', mode 'w' at 0x7f7aaed091e0>*)
A dummy logger implementation for debugging purposes that will just print to STDERR or whatever output stream it is given in the constructor.

external_data_file(*dummy1*, *dummy2*, *data*)

get_session_dir_name()

info(*text*)

alex.tools.autopath module

self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

Module contents

alex.utils package

Submodules

alex.utils.analytics module

alex.utils.audio module

alex.utils.audio_play module

alex.utils.autopath module

self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

alex.utils.cache module

class alex.utils.cache.**Counter**
Bases: dict

Mapping where default values are zero

alex.utils.cache.**get_persitent_cache_content**(*key*)

alex.utils.cache.**lfu_cache**(*maxsize=100*)
Least-frequently-used cache decorator.

Arguments to the cached function must be hashable. Cache performance statistics stored in f.hits and f.misses. Clear the cache with f.clear(). http://en.wikipedia.org/wiki/Least_Frequently_Used

alex.utils.cache.**lru_cache**(*maxsize=100*)
Least-recently-used cache decorator.

Arguments to the cached function must be hashable. Cache performance statistics stored in f.hits and f.misses. Clear the cache with f.clear(). http://en.wikipedia.org/wiki/Cache_algorithms#Least_Recently_Used

alex.utils.cache.**persistent_cache**(*method=False*, *file_prefix=''*, *file_suffix=''*)
Persistent cache decorator.

It grows indefinitely. Arguments to the cached function must be hashable. Cache performance statistics stored in f.hits and f.misses.

alex.utils.cache.**set_persitent_cache_content**(*key*, *value*)

alex.utils.caminfodb module

class alex.utils.caminfodb.**CamInfoDb**(*db_path*)
Bases: object

get_by_id(*rec_id*)

get_matching(*query*)

get_possible_values()

get_slots()

matches(*rec*, *query*)

alex.utils.config module

class alex.utils.config.**Config**(*file_name=None*, *project_root=False*, *config={}*)
Bases: object

Config handles configuration data necessary for all the components in Alex. It is implemented using a dictionary so that any component can use arbitrarily structured configuration data.

Before the configuration file is loaded, it is transformed as follows:

1. **'{cfg_abs_path}' as a string anywhere in the file is replaced by an** absolute path of the configuration files. This can be used to make the configuration file independent of the location of programs that use it.

DEFAULT_CFG_PPATH = u'resources/default.cfg'

config_replace(*p*, *s*, *d=None*)
Replace a pattern p with string s in the whole config (recursively) or in a part of the config given in d.

contains(**path*)
Check if configuration contains given keys (= path in config tree).

get(*i*, *default=None*)

getpath(*path*, *default=None*)

load(*file_name*)

classmethod load_configs(*config_flist=[]*, *use_default=True*, *log=True*, **init_args*, ***init_kwargs*)
Loads and merges configs from paths listed in 'config_flist'. Use this method instead of direct loading configs, as it takes care of not only merging them but also processing some options in a special way.

Arguments:

config_flist – list of paths to config files to load and merge; order matters (default: [])

use_default – whether to insert the default config ($ALEX/resources/default.cfg) at the beginning of 'config_flist' (default: True)

log – whether to log the resulting config using the system logger (default: True)

init_args – additional positional arguments will be passed to constructors for each config

init_kwargs – additional keyword arguments will be passed to constructors for each config

load_includes()

merge(*other*)
Merges self's config with other's config and saves it as a new self's config.

Keyword arguments:

- **other: a Config object whose configuration dictionary to merge** into self's one

unfold_lists(*pattern*, *unfold_id_key=None*, *part=[]*)
Unfold lists under keys matching the given pattern into several config objects, each containing one item. If pattern is None, all lists are expanded.

Stores a string representation of the individual unfolded values under the unfold_id_key if this parameter is set.

Only expands a part of the whole config hash (given by list of keys forming a path to this part) if the path parameter is set.

update(*new_config*, *config_dict=None*)
Updates the nested configuration dictionary by another, potentially also nested dictionary.

Keyword arguments:

- new_config: the new dictionary to update with
- config_dict: the config dictionary to be updated

alex.utils.config.**as_project_path**(*path*)

alex.utils.config.**callback_download_progress**(*blocks*, *block_size*, *total_size*)
callback function for urlretrieve that is called when connection is created and when once for each block

Parameters

- **blocks** – number of blocks transferred so far
- **block_size** – in bytes
- **total_size** – in bytes, can be -1 if server doesn't return it

alex.utils.config.**is_update_server_reachble**()

alex.utils.config.**load_as_module**(*path*, *force=False*, *encoding=u'UTF-8'*, *text_transforms=[]*)
Loads a file pointed to by 'path' as a Python module with minimal impact on the global program environment. The file name should end in '.py'.

Arguments: path – path towards the file force – whether to load the file even if its name does not end in '.py'

encoding – character encoding of the file text_transforms – collection of functions to be run on the original file text

Returns the loaded module object.

alex.utils.config.**online_update**(*file_name*)
This function can download file from a default server if it is not available locally. The default server location can be changed in the config file.

The original file name is transformed into absolute name using as_project_path function.

Parameters **file_name** – the file name which should be downloaded from the server

Returns a file name of the local copy of the file downloaded from the server

alex.utils.config.**set_online_update_server**(*server_name*)
Set the name of the online update server. This function can be used to change the server name from inside a config file.

Parameters **server_name** – the HTTP(s) path to the server and a location where the desired data reside.

Returns None

alex.utils.config.**to_project_path**(*path*)
Converts a relative or absoulute file system path to a path relative to project root.

alex.utils.cuda module

alex.utils.cuda.**cudasolve**(*A*, *b*, *tol=0.001*, *normal=False*, *regA=1.0*, *regI=0.0*)
Conjugate gradient solver for dense system of linear equations.

Ax = b

Returns: x = A^(-1)b

If the system is normal, then it solves

(regA*A'A +regI*I)x= b

Returns: x = (A'A +reg*I)^(-1)b

alex.utils.czech_stemmer module

Czech stemmer Copyright © 2010 Luís Gomes <luismsgomes@gmail.com>.

Ported from the Java implementation available at: http://members.unine.ch/jacques.savoy/clef/index.html

alex.utils.czech_stemmer.**cz_stem**(*l*, *aggressive=False*)

alex.utils.czech_stemmer.**cz_stem_word**(*word*, *aggressive=False*)

alex.utils.enums module

alex.utils.enums.**enum**(**sequential*, ***named*)
Useful for creating enumerations.

e.g.: DialogueType = enum(deterministic=0, statistical=1, mix=2)

alex.utils.env module

alex.utils.env.**root**()
Finds the root of the project and return it as string.

The root is the directory named alex.

alex.utils.excepthook module

Depending on the hook_type, ExceptionHook class adds various hooks how to catch exceptions.

class alex.utils.excepthook.**ExceptionHook**(*hook_type*, *logger=None*)
Bases: object

Singleton objects for registering various hooks for sys.exepthook. For registering a hook, use set_hook.

apply()
The object can be used to store settings for excepthook. a = ExceptionHook('log') # now it logs b = ExceptionHook('ipdb') # now it uses ipdb a.apply() # now it logs again

logger = None

classmethod set_hook(*hook_type=None*, *logger=None*)
Choose an exception hook from predefined functions.

hook_type: specify the name of the hook method

alex.utils.excepthook.**hook_decorator**(*f*)
Print the caution message when the decorated function raises an error.

alex.utils.excepthook.**ipdb_hook**(**args*, ***kwargs*)

alex.utils.excepthook.**log_and_ipdb_hook**(**args*, ***kwargs*)

alex.utils.excepthook.**log_hook**(**args*, ***kwargs*)

alex.utils.exceptions module

exception alex.utils.exceptions.**ConfigException**
Bases: alex.AlexException

exception alex.utils.exceptions.**SessionClosedException**
Bases: alex.AlexException

exception alex.utils.exceptions.**SessionLoggerException**
Bases: alex.AlexException

alex.utils.exdec module

alex.utils.exdec.**catch_ioerror**(*user_function*, *msg=''*)

alex.utils.filelock module

Context manager for locking on a file. Obtained from

http://www.evanfosmark.com/2009/01 /cross-platform-file-locking-support-in-python/,

licensed under BSD.

This is thought to work safely on NFS too, in contrast to fcntl.flock(). This is also thought to work safely over SMB and else, in contrast to fcntl.lockf(). For both issues, consult http://oilq.org/fr/node/13344.

Use as simply as

with FileLock(filename): <critical section for working with the file at 'filename'>

class alex.utils.filelock.**FileLock**(*file_name*, *timeout=10*, *delay=0.05*)
Bases: object

A file locking mechanism that has context-manager support so you can use it in a with statement. This should be relatively portable as it doesn't rely on msvcrt or fcntl for the locking.

acquire()
Acquire the lock, if possible. If the lock is in use, it check again every 'wait' seconds. It does this until it either gets the lock or exceeds 'timeout' number of seconds, in which case it throws an exception.

release()
Get rid of the lock by deleting the lockfile. When working in a 'with' statement, this method gets automatically called at the end.

exception alex.utils.filelock.**FileLockException**
Bases: exceptions.Exception

alex.utils.fs module

Filesystem utility functions.

class alex.utils.fs.**GrepFilter**(*stdin*, *stdout*, *breakchar=u'n'*)
Bases: multiprocessing.process.Process

add_listener(*regex*, *callback*)
Adds a listener to the output strings.

Arguments:

regex – the compiled regular expression to look for ('regex.search') in any piece of output

callback – a callable that is invoked for output where 'regex' was found. This will be called like this:

outputting &= callback(output_unicode_str)

That means, callback should take the unicode string argument containing what would have been output and return a boolean value which is True iff outputting should stop.

Returns the index of the listener for later reference.

flush(*force=True*)

remove_listener(*listener_idx*)

run()

write(*unistr*)

alex.utils.fs.**find**(*dir_*, *glob_*, *mindepth=2*, *maxdepth=6*, *ignore_globs=[]*, *ignore_paths=None*, *follow_symlinks=True*, *prune=False*, *rx=None*, *notrx=None*)

A simplified version of the GNU 'find' utility. Lists files with basename matching 'glob_' found in 'dir_' in depth between 'mindepth' and 'maxdepth'.

The 'ignore_globs' argument specifies a glob for basenames of files to be ignored. The 'ignore_paths' argument specifies a collection of real absolute pathnames that are pruned from the search. For efficiency reasons, it should be a set.

In the current implementation, the traversal resolves symlinks before the file name is checked. However, taking symlinks into account can be forbidden altogether by specifying 'follow_symlinks=False'. Cycles during the traversal are avoided.

- prune: whether to prune the subtree below a matching directory
- **rx: regexp to use as an additional matching criterion apart from 'glob_'**; the 're.match' function is used, as opposed to 're.find'
- notrx: like 'rx' but this specifies the regexp that must NOT match

The returned set of files consists of real absolute pathnames of those files.

alex.utils.fs.**normalise_path**(*path*)

Normalises a filesystem path using tilde expansion, absolutising and normalising the path, and resolving symlinks.

alex.utils.fs.**test_grep_filter**()

alex.utils.htk module

alex.utils.interface module

class alex.utils.interface.**Interface**

Bases: object

alex.utils.interface.**interface_method**(*f*)

alex.utils.lattice module

alex.utils.mfcc module

alex.utils.mproc module

Implements useful classes for handling multiprocessing implementation of the Alex system.

class alex.utils.mproc.**InstanceID**
Bases: object

This class provides unique ids to all instances of objects inheriting from this class.

get_instance_id(**args*, ***kw*)

instance_id = <Synchronized wrapper for c_int(0)>

lock = <Lock(owner=None)>

class alex.utils.mproc.**SystemLogger**(*output_dir*, *stdout_log_level='DEBUG'*, *stdout=True*, *file_log_level='DEBUG'*)
Bases: object

This is a multiprocessing-safe logger. It should be used by all components in Alex.

critical(**args*, ***kwargs*)

debug(**args*, ***kwargs*)

error(**args*, ***kwargs*)

exception(*message*)

formatter(**args*, ***kw*)
Format the message - pretty print

get_session_dir_name(**args*, ***kw*)
Return directory where all the call related files should be stored.

get_time_str()
Return current time in dashed ISO-like format.

It is useful in constructing file and directory names.

info(**args*, ***kwargs*)

levels = {'INFO': 20, 'CRITICAL': 40, 'EXCEPTION': 50, 'SYSTEM-LOG': 0, 'WARNING': 30, 'ERROR': 60, 'DE

lock = <RLock(None, 0)>

log(**args*, ***kw*)
Logs the message based on its level and the logging setting. Before writing into a logging file, it locks the file.

session_end(**args*, ***kw*)
WARNING: Deprecated Disables logging into the session-specific directory.

We better do not end a session because very often after the session_end() method is called there are still incoming messages. Therefore, it is better to wait for the session_start() method to set a new destination for the session log.

session_start(**args*, ***kw*)
Create a specific directory for logging a specific call.

NOTE: This is not completely safe. It can be called from several processes.

session_system_log(**args*, ***kwargs*)
This logs specifically only into the call-specific system log.

warning(**args*, ***kwargs*)

alex.utils.mproc.**async**(*func*)
A function decorator intended to make "func" run in a separate thread (asynchronously). Returns the created Thread object

E.g.: @async def task1():

do_something

@async def task2():

do_something_too

t1 = task1() t2 = task2() ... t1.join() t2.join()

alex.utils.mproc.**etime**(*name='Time'*, *min_t=0.3*)
This decorator measures the execution time of the decorated function.

alex.utils.mproc.**file_lock**(*file_name*)
Multiprocessing lock using files. Lock on a specific file.

alex.utils.mproc.**file_unlock**(*lock_file*)
Multiprocessing lock using files. Unlock on a specific file.

alex.utils.mproc.**global_lock**(*lock*)
This decorator makes the decorated function thread safe.

Keyword arguments: lock – a global variable pointing to the object to lock on

alex.utils.mproc.**local_lock**()
This decorator makes the decorated function thread safe.

For each function it creates a unique lock.

alex.utils.nose_plugins module

alex.utils.parsers module

class alex.utils.parsers.**CamTxtParser**(*lower=False*)
Bases: object

Parser of files of the following format: <<BOF>> [record]

[record]

... <<EOF>>

where [record] has the following format:

<<[record]>> [property name]([property value]) <</[record]>>

[property name] and [property value] are arbitrary strings

Any " or ' characters are stripped from the beginning and end of each [property value].

line_expr = <_sre.SRE_Pattern object at 0x7ff39347f030>

parse(*f_obj*)
Parse the given file and return list of dictionaries with parsed values.

Arguments: f_obj – filename of file or file object to be parsed

alex.utils.procname module

alex.utils.procname.**get_proc_name**()

alex.utils.procname.**set_proc_name**(*newname*)

alex.utils.rdb module

class alex.utils.rdb.**Rdb**(*port=4446*)
Bases: pdb.Pdb

do_c(*arg*)

do_cont(*arg*)

do_continue(*arg*)

alex.utils.sessionlogger module

class alex.utils.sessionlogger.**SessionLogger**
Bases: multiprocessing.process.Process

This is a multiprocessing-safe logger. It should be used by Alex to log information according the SDC 2010 XML format.

Date and times should also include time zone.

Times should be in seconds from the beginning of the dialogue.

cancel_join_thread()

run()

set_cfg(*cfg*)

set_close_event(*close_event*)

alex.utils.test_analytics module

alex.utils.test_fs module

Unit tests for alex.util.fs.

class alex.utils.test_fs.**TestFind**(*methodName='runTest'*)
Bases: unittest.case.TestCase

setUp()
Creates a playground of a directory tree. It looks like this: <testroot>/

- **a/**
  - aa/
  - ab/
  - **ac/**
    - **aca/**
      - acaa/
      - acab -> baaaa
- **b/**
  - **ba/**
    - **baa/**

- **baaa/**
  - baaaa -> daaa
  - **baaab/**
    - **baaaba/**
      - baaabaa/
      - baaabab -> ca

- **c/**
  - **ca/**
    - caa/
    - **cab/**
      - caba/
    - cac -> db
- **d/**
  - **da/**
    - **daa/**
      - daaa -> acab
  - db -> baaaba

tearDown()
Deletes the mock-up directory tree.

test_cycles()
Test the processing of cycles in the directory structure.

test_depth()
Tests mindepth and maxdepth.

test_globs()
Tests processing of the selection glob.

test_ignore_globs()
Test the functionality of ignore globs.

test_symlinks1()
Basic test for symlinks.

test_wrong_args()
Test for handling wrong arguments.

alex.utils.test_sessionlogger module

alex.utils.test_text module

class alex.utils.test_text.**TestString**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_parse_command()

test_split_by()

alex.utils.text module

class alex.utils.text.**Escaper**(*chars=u'‘"'*, *escaper=u'\'*, *re_flags=0*)
Bases: object

Creates a customised escaper for strings. The characters that need escaping, as well as the one used for escaping can be specified.

ESCAPED = 1

ESCAPER = 0

NORMAL = 2

annotate(*esced*)
Annotates each character of a text that has been escaped whether:
Escaper.ESCAPER - it is the escape character Escaper.ESCAPED - it is a character that was escaped Escaper.NORMAL - otherwise.
It is expected that only parts of the text may have actually been escaped.

Returns a list with the annotation values, co-indexed with characters of the input text.

escape(*text*)
Escapes the text using the parameters defined in the constructor.

static re_literal(*char*)
Escapes the character so that when it is used in a regexp, it matches itself.

static re_literal_list(*chars*)
Builds a [] group for a regular expression that matches exactly the characters specified.

unescape(*text*)
Unescapes the text using the parameters defined in the constructor.

alex.utils.text.**escape_special_characters_shell**(*text*, *characters=u'\"'*)
Simple function that tries to escape quotes. Not guaranteed to produce the correct result!! If that is needed, use the new ‘Escaper’ class.

alex.utils.text.**findall**(*text*, *char*, *start=0*, *end=-1*)

alex.utils.text.**min_edit_dist**(*target*, *source*)
Computes the min edit distance from target to source.

alex.utils.text.**min_edit_ops**(*target*, *source*, *cost=<function <lambda> at 0x7ff392c14758>*)
Computes the min edit operations from target to source.
Parameters
- **target** – a target sequence
- **source** – a source sequence
- **cost** – an expression for computing cost of the edit operations

Returns a tuple of (insertions, deletions, substitutions)

alex.utils.text.**parse_command**(*command*)
Parse the command name(var1=”val1”,...) into a dictionary structure:

E.g. call(destination=”1245”,opt=”X”) will be parsed into:
{ “__name__”: “call”, “destination”: “1245”, “opt”: “X”}
Return the parsed command in a dictionary.

alex.utils.text.**split_by**(*text*, *splitter*, *opening_parentheses=u'‘'*, *closing_parentheses=u'‘'*, *quotes=u'\"'*)
Splits the input text at each occurrence of the splitter only if it is not enclosed in parentheses.

text - the input text string splitter - multi-character string which is used to determine the position

of splitting of the text

opening_parentheses - an iterable of opening parentheses that has to be respected when splitting, e.g. "{(" (default: '')

closing_parentheses - an iterable of closing parentheses that has to be respected when splitting, e.g. "})" (default: '')

quotes - an iterable of quotes that have to come in pairs, e.g. '"'

alex.utils.text.**split_by_comma**(*text*)

alex.utils.token module

alex.utils.token.**get_token**(*cfg*)

alex.utils.ui module

alex.utils.ui.**getTerminalSize**()
Retrieves the size of the current terminal window.

Returns (None, None) in case of lack of success.

alex.utils.various module

alex.utils.various.**crop_to_finite**(*val*)

alex.utils.various.**flatten**(*list_*, *ltypes=(<type 'list'>, <type 'tuple'>)*)
Flatten nested list into a simple list.

alex.utils.various.**get_text_from_xml_node**(*node*)
Get text from all child nodes and concatenate it.

alex.utils.various.**group_by**(*objects*, *attrs*)
Groups 'objects' by the values of their attributes 'attrs'.

Returns a dictionary mapping from a tuple of attribute values to a list of objects with those attribute values.

class alex.utils.various.**nesteddict**
Bases: collections.defaultdict

walk()

alex.utils.various.**remove_dups_stable**(*l*)
Remove duplicates from a list but keep the ordering.

@return: Iterator over unique values in the list

alex.utils.various.**split_to_bins**(*A*, *S=4*)
Split the A array into bins of size N.

Module contents

class alex.utils.**DummyLogger**
Bases: object

alex.utils.**one**()

alex.utils.**script_path**(*fname*, **args*)
Return path relative to the directory of the given file, and join the additional path parts.
Args: fname (str): file used to determine the root directory args (list): additional path parts

2.1.2 Submodules

2.1.3 alex.autopath module

self cloning, automatic path configuration

copy this into any subdirectory of pypy from which scripts need to be run, typically all of the test subdirs. The idea is that any such script simply issues

import autopath

and this will make sure that the parent directory containing "pypy" is in sys.path.

If you modify the master "autopath.py" version (in pypy/tool/autopath.py) you can directly run it which will copy itself on all autopath.py files it finds under the pypy root directory.

This module always provides these attributes:

pypydir pypy root directory path this_dir directory where this autopath.py resides

2.1.4 Module contents

exception alex.**AlexException**
Bases: exceptions.Exception

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

CHAPTER 4

How to write documentation

- Quick cheatsheet for ReST and Sphinx
- More thorough documentation with code examples

[HTKBook] The HTK Book, version 3.4

a

alex, 114
alex.applications, 48
alex.applications.autopath, 48
alex.applications.exceptions, 48
alex.applications.PublicTransportInfoCS, 38
alex.applications.PublicTransportInfoCS.autopath, 36
alex.applications.PublicTransportInfoCS.crws_enums, 36
alex.applications.PublicTransportInfoCS.data, 33
alex.applications.PublicTransportInfoCS.data.add_cities_to_stops, 31
alex.applications.PublicTransportInfoCS.data.autopath, 32
alex.applications.PublicTransportInfoCS.data.database, 32
alex.applications.PublicTransportInfoCS.data.download_data, 32
alex.applications.PublicTransportInfoCS.data.get_cities_location, 32
alex.applications.PublicTransportInfoCS.data.ontology, 32
alex.applications.PublicTransportInfoCS.exceptions, 37
alex.applications.PublicTransportInfoCS.hcig, 33
alex.applications.PublicTransportInfoCS.hcig.autopath, 33
alex.applications.PublicTransportInfoCS.platform_info, 38
alex.applications.PublicTransportInfoCS.platform_info_test, 38
alex.applications.PublicTransportInfoCS.slu, 35
alex.applications.PublicTransportInfoCS.slu.add_to_bootstrap, 35
alex.applications.PublicTransportInfoCS.slu.autopath, 35
alex.applications.PublicTransportInfoCS.slu.consoli 35
alex.applications.PublicTransportInfoCS.slu.dailogr 34
alex.applications.PublicTransportInfoCS.slu.dailogr 33
alex.applications.PublicTransportInfoCS.slu.dailogr 34
alex.applications.PublicTransportInfoCS.slu.dainncl 34
alex.applications.PublicTransportInfoCS.slu.dainncl 34
alex.applications.PublicTransportInfoCS.slu.dainncl 34
alex.applications.PublicTransportInfoEN, 47
alex.applications.PublicTransportInfoEN.autopath, 44
alex.applications.PublicTransportInfoEN.data, 43
alex.applications.PublicTransportInfoEN.data.autopa 41
alex.applications.PublicTransportInfoEN.data.databa 41
alex.applications.PublicTransportInfoEN.data.downlo 41
alex.applications.PublicTransportInfoEN.data.expand 41
alex.applications.PublicTransportInfoEN.data.expand 41
alex.applications.PublicTransportInfoEN.data.expand 41
alex.applications.PublicTransportInfoEN.data.expand 42
alex.applications.PublicTransportInfoEN.data.expand 43
alex.applications.PublicTransportInfoEN.data.ontolo 43
alex.applications.PublicTransportInfoEN.data.prepro 40

alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.a2w, 39
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.a2w.cs, 39
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.a2w.cs.removerep, 40
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.read, 45
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.read.tectotempla, 46
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.read.yaml, 46
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.t2a, 44
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.t2a.addauxwords, 44
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.t2a.copyttree, 44
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.t2a.cs, 44
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.t2a.cs.addappos, 44
alex.applications.PublicTransportInfoEN.alex.components.nlg.tectotpl.block.t2a.cs.addauxver, 47
alex.applications.utils, 47
alex.applications.utils.weather, 47
alex.autopath, 114
alex.components, 84
alex.components.asr, 49
alex.components.asr.autopath, 49
alex.components.asr.common, 49
alex.components.asr.exceptions, 49
alex.components.dm, 54
alex.components.dm.autopath, 50
alex.components.dm.base, 50
alex.components.dm.common, 51
alex.components.dm.exceptions, 52
alex.components.dm.ontology, 52
alex.components.dm.pstate, 53
alex.components.dm.state, 54
alex.components.dm.tracker, 54
alex.components.hub, 56
alex.components.hub.calldb, 55
alex.components.hub.exceptions, 55
alex.components.hub.hub, 55
alex.components.hub.messages, 56
alex.components.nlg, 81
alex.components.nlg.autopath, 80
alex.components.nlg.exceptions, 81
alex.components.nlg.tectotpl, 80
alex.components.nlg.tectotpl.block, 70
alex.components.nlg.tectotpl.block.t2a.cs.addauxver, 57
alex.components.nlg.tectotpl.block.t2a.cs.addauxver, 57
alex.components.nlg.tectotpl.block.t2a.cs.addauxver, 57
alex.components.nlg.tectotpl.block.t2a.cs.addauxver, 57
alex.components.nlg.tectotpl.block.t2a.cs.addclausa, 59
alex.components.nlg.tectotpl.block.t2a.cs.addclausa, 60
alex.components.nlg.tectotpl.block.t2a.cs.addcoordp, 60
alex.components.nlg.tectotpl.block.t2a.cs.addparent, 60
alex.components.nlg.tectotpl.block.t2a.cs.addprepos, 61
alex.components.nlg.tectotpl.block.t2a.cs.addreflex, 61
alex.components.nlg.tectotpl.block.t2a.cs.addsentfi, 61
alex.components.nlg.tectotpl.block.t2a.cs.addsubcon, 61
alex.components.nlg.tectotpl.block.t2a.cs.addsuborc, 62
alex.components.nlg.tectotpl.block.t2a.cs.capitaliz, 62

alex.components.nlg.tectotpl.block.t2a.cs.deletesuperfluousauxs, 62
alex.components.nlg.tectotpl.block.t2a.cs.dropsubjpersprons, 62
alex.components.nlg.tectotpl.block.t2a.cs.generatepossessiveadjectives, 63
alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr, 63
alex.components.nlg.tectotpl.block.t2a.cs.imposecomplagr, 63
alex.components.nlg.tectotpl.block.t2a.cs.imposeconnagr, 63
alex.components.nlg.tectotpl.block.t2a.cs.imposeprontagr, 64
alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpredagr, 64
alex.components.nlg.tectotpl.block.t2a.cs.initcomponent, 64
alex.components.nlg.tectotpl.block.t2a.cs.markcomponent, 65
alex.components.nlg.tectotpl.block.t2a.cs.markcomponentsegories, 65
alex.components.nlg.tectotpl.block.t2a.cs.movecomponentswackernagel, 65
alex.components.nlg.tectotpl.block.t2a.cs.projectclausenumber, 66
alex.components.nlg.tectotpl.block.t2a.cs.revercomponentsnotsdependency, 66
alex.components.nlg.tectotpl.block.t2a.cs.vocalizeprepos, 66
alex.components.nlg.tectotpl.block.t2a.impose_agreement, 67
alex.components.nlg.tectotpl.block.t2t, 68
alex.components.nlg.tectotpl.block.util, 69
alex.components.nlg.tectotpl.block.util.copy_tree, 68
alex.components.nlg.tectotpl.block.util.eval, 68
alex.components.nlg.tectotpl.block.util.setglobal, 69
alex.components.nlg.tectotpl.block.write, 70
alex.components.nlg.tectotpl.block.write.base_writer, 69
alex.components.nlg.tectotpl.block.write.yaml, 69
alex.components.nlg.tectotpl.core, 77
alex.components.nlg.tectotpl.core.block, 70
alex.components.nlg.tectotpl.core.document, 70
alex.components.nlg.tectotpl.core.exception, 72
alex.components.nlg.tectotpl.core.log, 72
alex.components.nlg.tectotpl.core.node, 72
alex.components.nlg.tectotpl.core.run, 76
alex.components.nlg.tectotpl.core.util, 77
alex.components.nlg.tectotpl.tool, 80
alex.components.nlg.tectotpl.tool.cluster, 78
alex.components.nlg.tectotpl.tool.lexicon, 78
alex.components.nlg.tectotpl.tool.lexicon.cs, 77
alex.components.nlg.tectotpl.tool.ml, 78
alex.components.nlg.tools, 80
alex.components.nlg.tools.en, 80
alex.components.slu, 82
alex.components.slu.autopath, 81
alex.components.slu.exceptions, 82
alex.components.slu.templateclassifier, 82
alex.components.tts, 84
alex.components.tts.autopath, 83
alex.components.tts.base, 83
alex.components.tts.exceptions, 83
alex.components.tts.preprocessing, 83
alex.components.vad, 84
alex.components.vad.power, 84
alex.corpustools, 92
alex.corpustools.autopath, 84
alex.corpustools.cued, 85
alex.corpustools.cued2utt_da_pairs, 86
alex.corpustools.cued2wavaskey, 87
alex.corpustools.cuedda, 88
alex.corpustools.grammar_weighted, 88
alex.corpustools.merge_uttcns, 89
alex.corpustools.num_time_stats, 89
alex.corpustools.semscore, 90
alex.corpustools.srilm_ppl_filter, 90
alex.corpustools.text_norm_cs, 90
alex.corpustools.text_norm_en, 90
alex.corpustools.text_norm_es, 91
alex.corpustools.ufaldatabase, 91
alex.corpustools.wavaskey, 91
alex.ml, 97
alex.ml.bn, 93
alex.ml.bn.autopath, 92
alex.ml.bn.lbp, 92
alex.ml.ep, 95
alex.ml.ep.node, 94

alex.ml.ep.test, 95
alex.ml.ep.turn, 95
alex.ml.exceptions, 95
alex.ml.hypothesis, 96
alex.ml.lbp, 95
alex.ml.test_hypothesis, 97
alex.tests, 98
alex.tests.autopath, 97
alex.tests.test_mproc, 98
alex.tests.test_numpy_with_optimised_ATLAS, 98
alex.tools, 101
alex.tools.apirequest, 101
alex.tools.autopath, 101
alex.tools.mturk, 100
alex.tools.mturk.bin, 99
alex.tools.mturk.bin.autopath, 99
alex.tools.mturk.bin.mturk, 99
alex.tools.vad, 100
alex.tools.vad.autopath, 100
alex.utils, 113
alex.utils.autopath, 101
alex.utils.cache, 102
alex.utils.caminfodb, 102
alex.utils.config, 102
alex.utils.cuda, 104
alex.utils.czech_stemmer, 105
alex.utils.enums, 105
alex.utils.env, 105
alex.utils.excepthook, 105
alex.utils.exceptions, 105
alex.utils.exdec, 106
alex.utils.filelock, 106
alex.utils.fs, 106
alex.utils.interface, 107
alex.utils.mproc, 107
alex.utils.parsers, 109
alex.utils.procname, 109
alex.utils.rdb, 110
alex.utils.sessionlogger, 110
alex.utils.test_fs, 110
alex.utils.test_text, 111
alex.utils.text, 112
alex.utils.token, 113
alex.utils.ui, 113
alex.utils.various, 113

A

A (class in alex.components.nlg.tectotpl.core.node), 72
A (class in alex.corpustools.grammar_weighted), 88
acquire() (alex.utils.filelock.FileLock method), 106
add() (alex.ml.hypothesis.ConfusionNetwork method), 96
add() (alex.ml.hypothesis.NBList method), 97
add_aux_anodes() (alex.components.nlg.tectotpl.core.node.T method), 75
add_cities_to_stops() (in module alex.applications.PublicTransportInfoCS.data.add_cities_to_stops), 31
add_comma_node() (alex.components.nlg.tectotpl.block.t2a.cs.addappositionpunct.AddAppositionPunct method), 58
add_comma_node() (alex.components.nlg.tectotpl.block.t2a.cs.addcoordpunct.AddCoordPunct method), 60
add_dependency() (alex.components.nlg.tectotpl.tool.cluster.Job method), 79
add_layer() (alex.ml.bn.lbp.LBP method), 93
add_layers() (alex.ml.bn.lbp.LBP method), 93
add_listener() (alex.utils.fs.GrepFilter method), 106
add_merge() (alex.ml.hypothesis.ConfusionNetwork method), 96
add_nodes() (alex.ml.bn.lbp.LBP method), 93
add_other() (alex.ml.hypothesis.NBList method), 97
add_parenthesis_node() (alex.components.nlg.tectotpl.block.t2a.cs.addparentheses.AddParentheses method), 60
add_slot_values_from_database() (in module alex.applications.PublicTransportInfoCS.data.ontology), 32
add_slot_values_from_database() (in module alex.applications.PublicTransportInfoEN.data.ontology), 43
AddAppositionPunct (class in alex.components.nlg.tectotpl.block.t2a.cs.addappositionpunct), 58
AddAuxVerbCompoundFuture (class in alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundfuture), 58
AddAuxVerbCompoundPassive (class in alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpa 59
AddAuxVerbCompoundPast (class in alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpa 59
AddAuxVerbConditional (class in alex.components.nlg.tectotpl.block.t2a.cs.addauxverbconditional) 59
AddAuxVerbModal (class in alex.components.nlg.tectotpl.block.t2a.cs.addauxverbmodal), 59
AddAuxWords (class in alex.components.nlg.tectotpl.block.t2a.addauxwords), 67
AddClausalExpletives (class in alex.components.nlg.tectotpl.block.t2a.cs.addclausalexpletives), 59
AddClausalPunct (class in alex.components.nlg.tectotpl.block.t2a.cs.addclausalpunct), 60
AddCoordPunct (class in alex.components.nlg.tectotpl.block.t2a.cs.addcoordpunct), 60
addOthers() (alex.ml.ep.node.GroupingNode method), 94
AddParentheses (class in alex.components.nlg.tectotpl.block.t2a.cs.addparentheses), 60
AddPrepositions (class in alex.components.nlg.tectotpl.block.t2a.cs.addprepositions), 61
AddReflexiveParticles (class in alex.components.nlg.tectotpl.block.t2a.cs.addreflexiveparticles), 61
AddSentFinalPunct (class in alex.components.nlg.tectotpl.block.t2a.cs.addsentfinalpunct), 61
AddSubconjs (class in alex.components.nlg.tectotpl.block.t2a.cs.addsubconjs), 61
AddSubordClausePunct (class in alex.components.nlg.tectotpl.block.t2a.cs.addsubordclausepunct),

62
alex (module), 114
alex.applications (module), 48
alex.applications.autopath (module), 48
alex.applications.exceptions (module), 48
alex.applications.PublicTransportInfoCS (module), 38
alex.applications.PublicTransportInfoCS.autopath (module), 36
alex.applications.PublicTransportInfoCS.crws_enums (module), 36
alex.applications.PublicTransportInfoCS.data (module), 33
alex.applications.PublicTransportInfoCS.data.add_cities_to_stops (module), 31
alex.applications.PublicTransportInfoCS.data.autopath (module), 32
alex.applications.PublicTransportInfoCS.data.database (module), 32
alex.applications.PublicTransportInfoCS.data.download_data (module), 32
alex.applications.PublicTransportInfoCS.data.get_cities_location (module), 32
alex.applications.PublicTransportInfoCS.data.ontology (module), 32
alex.applications.PublicTransportInfoCS.exceptions (module), 37
alex.applications.PublicTransportInfoCS.hclg (module), 33
alex.applications.PublicTransportInfoCS.hclg.autopath (module), 33
alex.applications.PublicTransportInfoCS.platform_info (module), 38
alex.applications.PublicTransportInfoCS.platform_info_test (module), 38
alex.applications.PublicTransportInfoCS.slu (module), 35
alex.applications.PublicTransportInfoCS.slu.add_to_bootstrap (module), 35
alex.applications.PublicTransportInfoCS.slu.autopath (module), 35
alex.applications.PublicTransportInfoCS.slu.consolidate_keyfiles (module), 35
alex.applications.PublicTransportInfoCS.slu.dailogregclassifier (module), 34
alex.applications.PublicTransportInfoCS.slu.dailogregclassifier.autopath (module), 33
alex.applications.PublicTransportInfoCS.slu.dailogregclassifier.download_models (module), 34
alex.applications.PublicTransportInfoCS.slu.dainnclassifier (module), 34
alex.applications.PublicTransportInfoCS.slu.dainnclassifier.autopath (module), 34
alex.applications.PublicTransportInfoCS.slu.dainnclassifier.download_models (module), 34
alex.applications.PublicTransportInfoEN (module), 47
alex.applications.PublicTransportInfoEN.autopath (module), 44
alex.applications.PublicTransportInfoEN.data (module), 43
alex.applications.PublicTransportInfoEN.data.autopath (module), 41
alex.applications.PublicTransportInfoEN.data.database (module), 41
alex.applications.PublicTransportInfoEN.data.download_data (module), 41
alex.applications.PublicTransportInfoEN.data.expand_boroughs_script (module), 41
alex.applications.PublicTransportInfoEN.data.expand_cities_script (module), 41
alex.applications.PublicTransportInfoEN.data.expand_states_script (module), 41
alex.applications.PublicTransportInfoEN.data.expand_stops_script (module), 42
alex.applications.PublicTransportInfoEN.data.expand_streets_script (module), 43
alex.applications.PublicTransportInfoEN.data.ontology (module), 43
alex.applications.PublicTransportInfoEN.data.preprocessing (module), 40
alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_ (module), 39
alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv (module), 39
alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_street (module), 40
alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_cs (module), 40
alex.applications.PublicTransportInfoEN.directions (module), 45
alex.applications.PublicTransportInfoEN.exceptions (module), 46
alex.applications.PublicTransportInfoEN.site_preprocessing (module), 46
alex.applications.PublicTransportInfoEN.slu (module), 44
alex.applications.PublicTransportInfoEN.slu.add_to_bootstrap (module), 44
alex.applications.PublicTransportInfoEN.slu.autopath (module), 44
alex.applications.PublicTransportInfoEN.slu.consolidate_keyfiles (module), 44
alex.applications.PublicTransportInfoEN.slu.query_google (module), 44
alex.applications.PublicTransportInfoEN.time_zone (module), 47
alex.applications.utils (module), 47
alex.applications.utils.weather (module), 47
alex.autopath (module), 114

alex.components (module), 84
alex.components.asr (module), 49
alex.components.asr.autopath (module), 49
alex.components.asr.common (module), 49
alex.components.asr.exceptions (module), 49
alex.components.dm (module), 54
alex.components.dm.autopath (module), 50
alex.components.dm.base (module), 50
alex.components.dm.common (module), 51
alex.components.dm.exceptions (module), 52
alex.components.dm.ontology (module), 52
alex.components.dm.pstate (module), 53
alex.components.dm.state (module), 54
alex.components.dm.tracker (module), 54
alex.components.hub (module), 56
alex.components.hub.calldb (module), 55
alex.components.hub.exceptions (module), 55
alex.components.hub.hub (module), 55
alex.components.hub.messages (module), 56
alex.components.nlg (module), 81
alex.components.nlg.autopath (module), 80
alex.components.nlg.exceptions (module), 81
alex.components.nlg.tectotpl (module), 80
alex.components.nlg.tectotpl.block (module), 70
alex.components.nlg.tectotpl.block.a2w (module), 57
alex.components.nlg.tectotpl.block.a2w.cs (module), 57
alex.components.nlg.tectotpl.block.a2w.cs.concatenatetokens (module), 57
alex.components.nlg.tectotpl.block.a2w.cs.removerepeatedtokens (module), 57
alex.components.nlg.tectotpl.block.read (module), 58
alex.components.nlg.tectotpl.block.read.tectotemplates (module), 57
alex.components.nlg.tectotpl.block.read.yaml (module), 58
alex.components.nlg.tectotpl.block.t2a (module), 68
alex.components.nlg.tectotpl.block.t2a.addauxwords (module), 67
alex.components.nlg.tectotpl.block.t2a.copyttree (module), 67
alex.components.nlg.tectotpl.block.t2a.cs (module), 67
alex.components.nlg.tectotpl.block.t2a.cs.addappositionpunct (module), 58
alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundfuture (module), 58
alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpassive (module), 59
alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpast (module), 59
alex.components.nlg.tectotpl.block.t2a.cs.addauxverbconditional (module), 59
alex.components.nlg.tectotpl.block.t2a.cs.addauxverbmodal (module), 59
alex.components.nlg.tectotpl.block.t2a.cs.addclausalexpletives (module), 59
alex.components.nlg.tectotpl.block.t2a.cs.addclausalpunct (module), 60
alex.components.nlg.tectotpl.block.t2a.cs.addcoordpunct (module), 60
alex.components.nlg.tectotpl.block.t2a.cs.addparentheses (module), 60
alex.components.nlg.tectotpl.block.t2a.cs.addprepositions (module), 61
alex.components.nlg.tectotpl.block.t2a.cs.addreflexiveparticles (module), 61
alex.components.nlg.tectotpl.block.t2a.cs.addsentfinalpunct (module), 61
alex.components.nlg.tectotpl.block.t2a.cs.addsubconjs (module), 61
alex.components.nlg.tectotpl.block.t2a.cs.addsubordclausepunct (module), 62
alex.components.nlg.tectotpl.block.t2a.cs.capitalizesentstart (module), 62
alex.components.nlg.tectotpl.block.t2a.cs.deletesuperfluousauxs (module), 62
alex.components.nlg.tectotpl.block.t2a.cs.dropsubjpersprons (module), 62
alex.components.nlg.tectotpl.block.t2a.cs.generatepossessiveadjectives (module), 63
alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr (module), 63
alex.components.nlg.tectotpl.block.t2a.cs.imposecomplagr (module), 63
alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr (module), 63
alex.components.nlg.tectotpl.block.t2a.cs.imposerelpronagr (module), 64
alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpredagr (module), 64
alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat (module), 64
alex.components.nlg.tectotpl.block.t2a.cs.marksubject (module), 65
alex.components.nlg.tectotpl.block.t2a.cs.markverbalcategories (module), 65
alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowackernagel (module), 65
alex.components.nlg.tectotpl.block.t2a.cs.projectclausenumber (module), 66
alex.components.nlg.tectotpl.block.t2a.cs.reversenumbernoundependency (module), 66
alex.components.nlg.tectotpl.block.t2a.cs.vocalizeprepos (module), 66
alex.components.nlg.tectotpl.block.t2a.imposeagreement (module), 67
alex.components.nlg.tectotpl.block.t2t (module), 68
alex.components.nlg.tectotpl.block.util (module), 69

alex.components.nlg.tectotpl.block.util.copytree (module), 68
alex.components.nlg.tectotpl.block.util.eval (module), 68
alex.components.nlg.tectotpl.block.util.setglobal (module), 69
alex.components.nlg.tectotpl.block.write (module), 70
alex.components.nlg.tectotpl.block.write.basewriter (module), 69
alex.components.nlg.tectotpl.block.write.yaml (module), 69
alex.components.nlg.tectotpl.core (module), 77
alex.components.nlg.tectotpl.core.block (module), 70
alex.components.nlg.tectotpl.core.document (module), 70
alex.components.nlg.tectotpl.core.exception (module), 72
alex.components.nlg.tectotpl.core.log (module), 72
alex.components.nlg.tectotpl.core.node (module), 72
alex.components.nlg.tectotpl.core.run (module), 76
alex.components.nlg.tectotpl.core.util (module), 77
alex.components.nlg.tectotpl.tool (module), 80
alex.components.nlg.tectotpl.tool.cluster (module), 78
alex.components.nlg.tectotpl.tool.lexicon (module), 78
alex.components.nlg.tectotpl.tool.lexicon.cs (module), 77
alex.components.nlg.tectotpl.tool.ml (module), 78
alex.components.nlg.tools (module), 80
alex.components.nlg.tools.en (module), 80
alex.components.slu (module), 82
alex.components.slu.autopath (module), 81
alex.components.slu.exceptions (module), 82
alex.components.slu.templateclassifier (module), 82
alex.components.tts (module), 84
alex.components.tts.autopath (module), 83
alex.components.tts.base (module), 83
alex.components.tts.exceptions (module), 83
alex.components.tts.preprocessing (module), 83
alex.components.vad (module), 84
alex.components.vad.power (module), 84
alex.corpustools (module), 92
alex.corpustools.autopath (module), 84
alex.corpustools.cued (module), 85
alex.corpustools.cued2utt_da_pairs (module), 86
alex.corpustools.cued2wavaskey (module), 87
alex.corpustools.cuedda (module), 88
alex.corpustools.grammar_weighted (module), 88
alex.corpustools.merge_uttcns (module), 89
alex.corpustools.num_time_stats (module), 89
alex.corpustools.semscore (module), 90
alex.corpustools.srilm_ppl_filter (module), 90
alex.corpustools.text_norm_cs (module), 90
alex.corpustools.text_norm_en (module), 90
alex.corpustools.text_norm_es (module), 91
alex.corpustools.ufaldatabase (module), 91
alex.corpustools.wavaskey (module), 91
alex.ml (module), 97
alex.ml.bn (module), 93
alex.ml.bn.autopath (module), 92
alex.ml.bn.lbp (module), 92
alex.ml.ep (module), 95
alex.ml.ep.node (module), 94
alex.ml.ep.test (module), 95
alex.ml.ep.turn (module), 95
alex.ml.exceptions (module), 95
alex.ml.hypothesis (module), 96
alex.ml.lbp (module), 95
alex.ml.test_hypothesis (module), 97
alex.tests (module), 98
alex.tests.autopath (module), 97
alex.tests.test_mproc (module), 98
alex.tests.test_numpy_with_optimised_ATLAS (module), 98
alex.tools (module), 101
alex.tools.apirequest (module), 101
alex.tools.autopath (module), 101
alex.tools.mturk (module), 100
alex.tools.mturk.bin (module), 99
alex.tools.mturk.bin.autopath (module), 99
alex.tools.mturk.bin.mturk (module), 99
alex.tools.vad (module), 100
alex.tools.vad.autopath (module), 100
alex.utils (module), 113
alex.utils.autopath (module), 101
alex.utils.cache (module), 102
alex.utils.caminfodb (module), 102
alex.utils.config (module), 102
alex.utils.cuda (module), 104
alex.utils.czech_stemmer (module), 105
alex.utils.enums (module), 105
alex.utils.env (module), 105
alex.utils.excepthook (module), 105
alex.utils.exceptions (module), 105
alex.utils.exdec (module), 106
alex.utils.filelock (module), 106
alex.utils.fs (module), 106
alex.utils.interface (module), 107
alex.utils.mproc (module), 107
alex.utils.parsers (module), 109
alex.utils.procname (module), 109
alex.utils.rdb (module), 110
alex.utils.sessionlogger (module), 110
alex.utils.test_fs (module), 110
alex.utils.test_text (module), 111
alex.utils.text (module), 112
alex.utils.token (module), 113
alex.utils.ui (module), 113
alex.utils.various (module), 113
AlexException, 114
all_to_lower() (in module alex.applications.PublicTransportInfoEN.data.expand_boroughs_41

all_to_lower() (in module alex.applications.PublicTransportInfoEN.data.expand_cities_script), 41
Alternative (class in alex.corpustools.grammar_weighted), 88
annotate() (alex.utils.text.Escaper method), 112
anodes (alex.components.nlg.tectotpl.core.node.T attribute), 76
APIRequest (class in alex.tools.apirequest), 101
append() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_script), 42
apply() (alex.utils.excepthook.ExceptionHook method), 105
apply_to() (alex.components.nlg.tectotpl.core.run.Scenario method), 76
are_in_coord_clauses() (alex.components.nlg.tectotpl.block.t2a.cs.addsubordclausepunct.AddSubordClausePunct method), 62
as_list() (in module alex.components.nlg.tectotpl.core.util), 77
as_project_path() (in module alex.utils.config), 103
as_terminal() (in module alex.corpustools.grammar_weighted), 89
as_weight_tuple() (in module alex.corpustools.grammar_weighted), 89
asr_factory() (in module alex.components.asr.common), 49
ASRException, 49
asrhyp (alex.corpustools.cued2utt_da_pairs.TurnRecord attribute), 86
ASRHyp (class in alex.components.hub.messages), 56
async() (in module alex.utils.mproc), 108
atree (alex.components.nlg.tectotpl.core.document.Zone attribute), 71
attrib (alex.components.nlg.tectotpl.core.node.A attribute), 72
attrib (alex.components.nlg.tectotpl.core.node.EffectiveRelations attribute), 73
attrib (alex.components.nlg.tectotpl.core.node.InClause attribute), 73
attrib (alex.components.nlg.tectotpl.core.node.N attribute), 74
attrib (alex.components.nlg.tectotpl.core.node.Node attribute), 74
attrib (alex.components.nlg.tectotpl.core.node.Ordered attribute), 75
attrib (alex.components.nlg.tectotpl.core.node.P attribute), 75
attrib (alex.components.nlg.tectotpl.core.node.T attribute), 76
audio (alex.corpustools.cued2utt_da_pairs.TurnRecord attribute), 86
aux_anodes (alex.components.nlg.tectotpl.core.node.T attribute), 76
AUX_PAST_FORMS (alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpast.AddAuxVerbCompoundPast attribute), 59
average_same_stops() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.us_c 40
average_same_stops() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.mta_ 39
average_same_stops() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops 40

B

BASE_DIST_LIMIT (alex.components.nlg.tectotpl.block.t2a.cs.deletesuper attribute), 62
BaseWriter (class in alex.components.nlg.tectotpl.block.write.basewriter), 69
BEDS (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
Block (class in alex.components.nlg.tectotpl.core.block), 70
BP (class in alex.ml.bn.lbp), 92
BPError, 92
bundle (alex.components.nlg.tectotpl.core.document.Zone attribute), 71
Bundle (class in alex.components.nlg.tectotpl.core.document), 70

C

callback_download_progress() (in module alex.utils.config), 104
CallDB (class in alex.components.hub.calldb), 55
CamInfoDb (class in alex.utils.caminfodb), 102
CamTxtParser (class in alex.utils.parsers), 109
cancel_join_thread() (alex.utils.sessionlogger.SessionLogger method), 110
CapitalizeSentStart (class in alex.components.nlg.tectotpl.block.t2a.cs.capitalizesentstart), 62
catch_ioerror() (in module alex.utils.exdec), 106
clamp_01() (in module alex.corpustools.grammar_weighted), 89
clear_layers() (alex.ml.bn.lbp.LBP method), 93
clear_nodes() (alex.ml.bn.lbp.LBP method), 93
CLIENTEXCEPTION_CODE (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
clitic_order() (alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowacke method), 65
close_database() (alex.components.hub.calldb.CallDB method), 55
COMBFLAGS (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
Command (class in alex.components.hub.messages), 56

compl_nodes (alex.components.nlg.tectotpl.core.node.T attribute), 76
ConcatenateTokens (class in alex.components.nlg.tectotpl.block.a2w.cs.concatenatetokens), 57
Config (class in alex.utils.config), 102
config_replace() (alex.utils.config.Config method), 103
ConfigException, 105
ConfusionNetwork (class in alex.ml.hypothesis), 96
ConfusionNetworkException, 96
ConstChangeGoal (class in alex.ml.ep.node), 94
contains() (alex.utils.config.Config method), 103
continued_paren_left() (alex.components.nlg.tectotpl.block.t2a.cs.addparentheses.AddParentheses method), 60
continued_paren_right() (alex.components.nlg.tectotpl.block.t2a.cs.addparentheses.AddParentheses method), 60
COOR (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
copy_subtree() (alex.components.nlg.tectotpl.block.t2a.copyttree.CopyTTree method), 67
copy_subtree() (alex.components.nlg.tectotpl.block.util.copytree.CopyTree method), 68
CopyTree (class in alex.components.nlg.tectotpl.block.util.copytree), 68
CopyTTree (class in alex.components.nlg.tectotpl.block.t2a.copyttree), 67
coref_gram_nodes (alex.components.nlg.tectotpl.core.node.T attribute), 76
coref_text_nodes (alex.components.nlg.tectotpl.core.node.T attribute), 76
Counter (class in alex.utils.cache), 102
counter_weight() (in module alex.corpustools.grammar_weighted), 89
CRCONST (class in alex.applications.PublicTransportInfoCS.crws_enums), 36
create_atree() (alex.components.nlg.tectotpl.core.document.Bundle method), 71
create_bundle() (alex.components.nlg.tectotpl.core.document.Document method), 71
create_child() (alex.components.nlg.tectotpl.core.node.Node method), 74
create_ntree() (alex.components.nlg.tectotpl.core.document.Bundle method), 71
create_ptree() (alex.components.nlg.tectotpl.core.document.Bundle method), 71
create_tree() (alex.components.nlg.tectotpl.core.document.Zone method), 71
create_ttree() (alex.components.nlg.tectotpl.core.document.Zone method), 71
create_zone() (alex.components.nlg.tectotpl.core.document.Bundle method), 70
critical() (alex.utils.mproc.SystemLogger method), 108
crop_to_finite() (in module alex.utils.various), 113
CRWSPlatformInfo (class in alex.applications.PublicTransportInfoCS.platform_info), 38
cudasolve() (in module alex.utils.cuda), 104
cued_da (alex.corpustools.cued2utt_da_pairs.TurnRecord attribute), 86
cued_dahyp (alex.corpustools.cued2utt_da_pairs.TurnRecord attribute), 86
CUEDDialogueAct (class in alex.corpustools.cuedda), 88
CuedDialogueActError, 82
CUEDSlot (class in alex.corpustools.cuedda), 88
cz_stem() (in module alex.utils.czech_stemmer), 105
cz_stem_word() (in module alex.utils.czech_stemmer), 105

D

da_in() (alex.components.dm.base.DialogueManager method), 50
da_out() (alex.components.dm.base.DialogueManager method), 50
DAIKernelException, 82
DAILRException, 82
DataException, 72
debug() (alex.utils.mproc.SystemLogger method), 108
decide() (alex.components.vad.power.PowerVAD method), 84
DEFAULT_CFG_PPATH (alex.utils.config.Config attribute), 103
DEFAULT_CORES (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
default_extension (alex.components.nlg.tectotpl.block.write.yaml.YAML attribute), 69
DEFAULT_HEADER (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
DEFAULT_MEMORY (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
DEGREE (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorp attribute), 64
DELAY_CD (alex.applications.PublicTransportInfoCS.crws_enums.CRCO attribute), 36
DELAY_INTERN (alex.applications.PublicTransportInfoCS.crws_enums.C attribute), 36
DELAY_INTERN_EXT (alex.applications.PublicTransportInfoCS.crws_en attribute), 36
DELAY_TELMAX1 (alex.applications.PublicTransportInfoCS.crws_enums attribute), 36
DELAY_ZSR (alex.applications.PublicTransportInfoCS.crws_enums.CRC attribute), 36
DeleteSuperfluousAuxs (class in alex.components.nlg.tectotpl.block.t2a.cs.deletesuperfluousauxs), 62
DELTAMAX (in module alex.applications.PublicTransportInfoCS.crws_enums), 36

DEONTMOD_2_MODAL (alex.components.nlg.tectotpl.block.t2a.cs.addauxmodalverbs attribute), 59
DEP_TABLE (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
DeterministicDiscriminativeDialogueStateException, 52
DialogueActConfusionNetworkException, 82
DialogueActException, 82
DialogueActItemException, 82
DialogueActNBListException, 82
DialogueManager (class in alex.components.dm.base), 50
DialogueManagerException, 52
DialoguePolicy (class in alex.components.dm.base), 50
DialoguePolicyException, 52
DialogueState (class in alex.components.dm.base), 50
DialogueStateException, 52
DIR_PREFIX (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
Directions (class in alex.applications.PublicTransportInfoEN.directions), 45
DirectionsFinder (class in alex.applications.PublicTransportInfoEN.directions), 45
DiscreteValue (class in alex.components.dm.base), 51
DIST_LIMIT (alex.components.nlg.tectotpl.block.t2a.cs.deletesuperfluousexpletives attribute), 62
dm_factory() (in module alex.components.dm.common), 51
DMDA (class in alex.components.hub.messages), 56
DMException, 52
do_c() (alex.utils.rdb.Rdb method), 110
do_cont() (alex.utils.rdb.Rdb method), 110
do_continue() (alex.utils.rdb.Rdb method), 110
document (alex.components.nlg.tectotpl.core.document.Bundle attribute), 70
document (alex.components.nlg.tectotpl.core.document.Zone attribute), 71
document (alex.components.nlg.tectotpl.core.node.Node attribute), 74
Document (class in alex.components.nlg.tectotpl.core.document), 71
drop_anode() (alex.components.nlg.tectotpl.block.t2a.cs.dropsubjperspron method), 62
DropSubjPersProns (class in alex.components.nlg.tectotpl.block.t2a.cs.dropsubjperspron), 62
DummyDialoguePolicyException, 52
DummyLogger (class in alex.tools.apirequest), 101
DummyLogger (class in alex.utils), 113

E

EffectiveRelations (class in alex.components.nlg.tectotpl.core.node), 73
end_dialogue() (alex.components.dm.base.DialogueManager method), 50
enum() (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
enum() (in module alex.utils.enums), 105
error() (alex.utils.mproc.SystemLogger method), 108
escape() (alex.utils.text.Escaper method), 112
escape_special_characters_shell() (in module alex.utils.text), 112
ESCAPED (alex.utils.text.Escaper attribute), 112
ESCAPER (alex.utils.text.Escaper attribute), 112
Escaper (class in alex.utils.text), 112
etime() (in module alex.utils.mproc), 109
Eval (class in alex.components.nlg.tectotpl.block.util.eval), 68
every_word_for_number() (in module alex.components.nlg.tools.en), 80
exception() (alex.utils.mproc.SystemLogger method), 108
EXCEPTIONEXCLUSION_CD (alex.applications.PublicTransportInfoCS.crws_enums.CRCONS attribute), 36
ExceptionHook (class in alex.utils.excepthook), 105
exclude_by_dict() (in module alex.corpustools.text_norm_cs), 90
exclude_by_dict() (in module alex.corpustools.text_norm_en), 90
exclude_by_dict() (in module alex.corpustools.text_norm_es), 91
EXFUNCTIONRESULT (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
exit_status (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
expand() (in module alex.applications.PublicTransportInfoEN.site_preproce 46
expand_place() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
expand_stop() (in module alex.applications.PublicTransportInfoEN.site_preprocessing), 46
explain() (alex.components.dm.base.DiscreteValue method), 51
explain() (alex.ml.ep.node.GroupingNode method), 94
explain() (alex.ml.ep.node.Node method), 94
extend() (alex.ml.hypothesis.ConfusionNetwork method), 96
external_data_file() (alex.tools.apirequest.DummyLogger method), 101
extract_fields() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.mta_ 39

extract_fields() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment), 40
extract_stops() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment), 40
extract_trns_sems() (in module alex.corpustools.cued2utt_da_pairs), 86
extract_trns_sems_from_file() (in module alex.corpustools.cued2utt_da_pairs), 86

F

FCS (in module alex.applications.PublicTransportInfoCS.crwsl), 36
FFNNException, 95
file_check() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_names), 42
file_lock() (in module alex.utils.mproc), 109
file_stream() (in module alex.components.nlg.tectotpl.core.util), 77
file_unlock() (in module alex.utils.mproc), 109
FileLock (class in alex.utils.filelock), 106
FileLockException, 106
find() (in module alex.utils.fs), 107
find_best_cn() (in module alex.corpustools.merge_uttcns), 89
find_eo1st_pos() (alex.components.nlg.tectotpl.block.t2a.cs.getvalitforms method), 65
find_logs() (in module alex.corpustools.cued), 85
find_platform_by_station() (alex.applications.PublicTransportInfoCS.platform_info.CRWSPlatformInfo method), 38
find_platform_by_train_name() (alex.applications.PublicTransportInfoCS.platform_info.CRWSPlatformInfo method), 38
find_wavs() (in module alex.corpustools.cued), 85
find_with_ignorelist() (in module alex.corpustools.cued), 85
findall() (in module alex.utils.text), 112
FINISH (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
first() (in module alex.components.nlg.tectotpl.core.util), 77
fix_ordinal() (in module alex.applications.PublicTransportInfoEN.site_preprocessing), 46
flatten() (in module alex.utils.various), 113
flush() (alex.utils.fs.GrepFilter method), 107
formatter() (alex.utils.mproc.SystemLogger method), 108
Frame (class in alex.components.hub.messages), 56
from_fact() (alex.ml.hypothesis.ConfusionNetwork class method), 96
from_fact() (alex.ml.hypothesis.Hypothesis class method), 96
from_fact() (alex.ml.hypothesis.NBList class method), 97

G

GENDER (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorp attribute), 64
GeneratePossessiveAdjectives (class in alex.components.nlg.tectotpl.block.t2a.cs.generatepossessiveadjec 63
get() (alex.components.dm.pstate.PDDiscrete method), 53
get() (alex.components.dm.pstate.PDDiscreteOther method), 54
get() (alex.utils.config.Config method), 103
get_all_zones() (alex.components.nlg.tectotpl.core.document.Bundle method), 70
get_anode() (alex.components.nlg.tectotpl.block.t2a.addauxwords.AddAux method), 67
get_anode() (alex.components.nlg.tectotpl.block.t2a.cs.addclausalexpletives method), 59
get_asr_type() (in module alex.components.asr.common), 49
get_attr() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_attr_list() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_aux_forms() (alex.components.nlg.tectotpl.block.t2a.addauxwords.Add method), 67
get_aux_forms() (alex.components.nlg.tectotpl.block.t2a.cs.addclausalexple method), 60
get_aux_forms() (alex.components.nlg.tectotpl.block.t2a.cs.addprepositions method), 61
get_aux_forms() (alex.components.nlg.tectotpl.block.t2a.cs.addsubconjs.Ad method), 61
get_best() (alex.components.dm.pstate.PDDiscreteBase method), 54
get_best() (alex.ml.hypothesis.NBList method), 97
get_by_id() (alex.utils.caminfodb.CamInfoDb method), 102
get_call_data_from_fs() (in module alex.corpustools.num_time_stats), 89
get_call_data_from_log() (in module alex.corpustools.num_time_stats), 89
get_children() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_city_for_stop() (in module alex.applications.PublicTransportInfoCS.data.add_cities_to_stops 31
get_clause_parent() (alex.components.nlg.tectotpl.block.t2a.cs.addsubordcl method), 62
get_clause_root() (alex.components.nlg.tectotpl.core.node.InClause method), 73

get_coap_members() (alex.components.nlg.tectotpl.core.node.EffectiveRelationsNode method), 73
get_column_index() (in module alex.applications.PublicTransportInfoEN.data.export_stops_to_csv), 42
get_column_index() (in module alex.applications.PublicTransportInfoEN.data.preprocessing_metro_to_csv), 39
get_column_index() (in module alex.applications.PublicTransportInfoEN.data.preprocessing_streets_to_csv), 40
get_column_index() (in module alex.applications.PublicTransportInfoEN.data.preprocessing_stops_to_csv), 40
get_compatible_vals() (alex.components.dm.ontology.Ontology method), 52
get_cued_da() (alex.corpustools.cuedda.CUEDDialogueAct method), 88
get_da() (alex.components.dm.base.DialoguePolicy method), 50
get_default_value() (alex.components.dm.ontology.Ontology method), 52
get_depth() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_deref_attr() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_descendants() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_directions() (alex.applications.PublicTransportInfoEN.directions.DirectionsFinder method), 45
get_directions() (alex.applications.PublicTransportInfoEN.directions.GoogleDirectionsFinder method), 45
get_distrib() (alex.components.dm.pstate.PDDiscrete method), 53
get_distrib() (alex.components.dm.pstate.PDDiscreteOther method), 54
get_dm_type() (in module alex.components.dm.common), 51
get_echildren() (alex.components.nlg.tectotpl.core.node.EffectiveRelationsNode method), 73
get_entropy() (alex.components.dm.pstate.PDDiscrete method), 53
get_entropy() (alex.components.dm.pstate.PDDiscreteOther method), 54
get_eparents() (alex.components.nlg.tectotpl.core.node.EffectiveRelationsNode method), 73
get_google_coords() (in module alex.applications.PublicTransportInfoCS.data.get_cities_location), 32
get_instance_id() (alex.utils.mproc.InstanceID method), 108
get_items() (alex.components.dm.pstate.PDDiscrete method), 53
get_items() (alex.components.dm.pstate.PDDiscreteOther method), 54
get_matching() (alex.utils.caminfodb.CamInfoDb method), 102
get_max() (alex.components.dm.pstate.PDDiscrete method), 53
get_max() (alex.components.dm.pstate.PDDiscreteOther method), 54
get_minimal_info() (alex.applications.PublicTransportInfoEN.directions.TransportDirections method), 46
get_next_node() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
get_node_by_id() (alex.components.nlg.tectotpl.core.document.Document method), 71
get_or_create_zone() (alex.components.nlg.tectotpl.core.document.Bundle method), 70
get_output_file_name() (alex.components.nlg.tectotpl.block.write.basewrite method), 69
get_persitent_cache_content() (in module alex.utils.cache), 102
get_possessive_adj_for() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
get_possible_values() (alex.utils.caminfodb.CamInfoDb method), 102
get_prev_node() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
get_prob() (alex.ml.hypothesis.ConfusionNetwork method), 96
get_proc_name() (in module alex.utils.procname), 109
get_ref_attr_list() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_referenced_ids() (alex.components.nlg.tectotpl.core.node.Node method), 74
get_script_text() (alex.components.nlg.tectotpl.tool.cluster.Job method), 79
get_session_dir_name() (alex.tools.apirequest.DummyLogger method), 101
get_session_dir_name() (alex.utils.mproc.SystemLogger method), 108
get_slots() (alex.utils.caminfodb.CamInfoDb method), 102
get_slots_and_values() (alex.corpustools.cuedda.CUEDDialogueAct method), 88
get_slots_being_confirmed() (alex.components.dm.base.DialogueState method), 50
get_slots_being_noninformed() (alex.components.dm.base.DialogueState method), 50
get_slots_being_requested() (alex.components.dm.base.DialogueState method), 51
get_text_from_xml_node() (in module alex.utils.various), 113

get_time() (alex.applications.PublicTransportInfoEN.time_zone.GoogleTimeFinder method), 47
get_time_str() (alex.components.hub.messages.Message method), 56
get_time_str() (alex.utils.mproc.SystemLogger method), 108
get_timestamp() (in module alex.corpustools.num_time_stats), 90
get_token() (in module alex.utils.token), 113
get_tree() (alex.components.nlg.tectotpl.core.document.Zone method), 71
get_ufal_da() (alex.corpustools.cuedda.CUEDDialogueAct method), 88
get_uri_stats() (alex.components.hub.calldb.CallDB method), 55
get_weather() (alex.applications.utils.weather.OpenWeatherMapWeatherFinder method), 47
get_weather() (alex.applications.utils.weather.WeatherFinder method), 47
get_zone() (alex.components.nlg.tectotpl.core.document.Bundle method), 70
getMostProbableValue() (alex.ml.ep.node.Node method), 94
getpath() (alex.utils.config.Config method), 103
getTerminalSize() (in module alex.utils.ui), 113
getTwoMostProbableValues() (alex.ml.ep.node.Node method), 94
global_lock() (in module alex.utils.mproc), 109
Goal (class in alex.ml.ep.node), 94
GoogleDirections (class in alex.applications.PublicTransportInfoEN.directions), 45
GoogleDirectionsFinder (class in alex.applications.PublicTransportInfoEN.directions), 45
GoogleRoute (class in alex.applications.PublicTransportInfoEN.directions), 45
GoogleRouteLeg (class in alex.applications.PublicTransportInfoEN.directions), 45
GoogleRouteLegStep (class in alex.applications.PublicTransportInfoEN.directions), 45
GoogleTimeFinder (class in alex.applications.PublicTransportInfoEN.time_zone), 47
gram_aspect (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_degcmp (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_deontmod (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_diathesis (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_dispmod (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_gender (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_indeftype (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_iterativeness (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_negation (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_number (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_numertype (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_person (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_politeness (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_resultative (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_sempos (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_tense (alex.components.nlg.tectotpl.core.node.T attribute), 76
gram_verbmod (alex.components.nlg.tectotpl.core.node.T attribute), 76
GrammarGen (class in alex.corpustools.grammar_weighted), 88
GrepFilter (class in alex.utils.fs), 106
group_by() (in module alex.utils.various), 113
group_by_city_and_state() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.us_c 40
group_by_name() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.mta_ 39
group_by_name() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops 40
GroupingGoal (class in alex.ml.ep.node), 94
GroupingNode (class in alex.ml.ep.node), 94

H

hack_stops() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
handle_boroughs() (in module alex.applications.PublicTransportInfoEN.data.expand_boroughs_ 41
handle_cities() (in module alex.applications.PublicTransportInfoEN.data.expand_cities_scrip 41

handle_compatibility() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_script), 42
handle_compatibility() (in module alex.applications.PublicTransportInfoEN.data.preprocessing_compatibility_script), 39
handle_csv() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_script), 42
handle_pronoun_je() (alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr.ImposePronZAgr method), 66
handle_states() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_script), 41
has_atree() (alex.components.nlg.tectotpl.core.document.Zone method), 71
has_expletive() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
has_ntree() (alex.components.nlg.tectotpl.core.document.Zone method), 71
has_ptree() (alex.components.nlg.tectotpl.core.document.Zone method), 71
has_synthetic_future() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
has_tree() (alex.components.nlg.tectotpl.core.document.Zone method), 72
has_ttree() (alex.components.nlg.tectotpl.core.document.Zone method), 72
has_zone() (alex.components.nlg.tectotpl.core.document.Bundle method), 70
hook_decorator() (in module alex.utils.excepthook), 105
host (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
Hub (class in alex.components.hub.hub), 55
hub_type (alex.components.hub.hub.Hub attribute), 55
HubException, 48
Hypothesis (class in alex.ml.hypothesis), 96

I

id (alex.components.nlg.tectotpl.core.node.Node attribute), 74
impose() (alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr.ImposeAttrAgr method), 63
impose() (alex.components.nlg.tectotpl.block.t2a.cs.imposecomplagr.ImposeComplAgr method), 63
impose() (alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr.ImposePronZAgr method), 64
impose() (alex.components.nlg.tectotpl.block.t2a.cs.imposerelpronagr.ImposeRelPronAgr method), 64
impose() (alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpredagr.ImposeSubjPredAgr method), 64
impose() (alex.components.nlg.tectotpl.block.t2a.imposeagreement.ImposeAgreement method), 68
ImposeAgreement (class in alex.components.nlg.tectotpl.block.t2a.imposeagreement), 67
ImposeAttrAgr (class in alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr), 63
ImposeComplAgr (class in alex.components.nlg.tectotpl.block.t2a.cs.imposecomplagr), 63
ImposePronZAgr (class in alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr), 63
ImposeRelPronAgr (class in alex.components.nlg.tectotpl.block.t2a.cs.imposerelpronagr), 64
ImposeSubjPredAgr (class in alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpredagr), 64
InClause (class in alex.components.nlg.tectotpl.core.node), 73
index_backref() (alex.components.nlg.tectotpl.core.document.Document method), 71
index_node() (alex.components.nlg.tectotpl.core.document.Document method), 71
inflect_conditional() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
info() (alex.tools.apirequest.DummyLogger method), 101
info() (alex.utils.mproc.SystemLogger method), 108
init_messages() (alex.ml.bn.lbp.LBP method), 93
init_readline() (alex.components.hub.hub.Hub method), 55
InitMorphcat (class in alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat), 64
insert_comma_between() (alex.components.nlg.tectotpl.block.t2a.cs.addsubordclausepunct. method), 62
instance_id (alex.utils.mproc.InstanceID attribute), 108
InstanceID (class in alex.utils.mproc), 107
Interface (class in alex.utils.interface), 107
interface_method() (in module alex.utils.interface), 107
ipdb_hook() (in module alex.utils.excepthook), 105
is_at_clause_boundary() (alex.components.nlg.tectotpl.block.t2a.cs.addcoor method), 60
is_before_punct() (alex.components.nlg.tectotpl.block.t2a.cs.addapposition method), 58
is_clause_in_quotes() (alex.components.nlg.tectotpl.block.t2a.cs.addclausal method), 60
is_clitic() (alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowackerna method), 66
is_coap_root() (alex.components.nlg.tectotpl.core.node.A method), 73
is_coap_root() (alex.components.nlg.tectotpl.core.node.EffectiveRelations method), 73

is_coap_root() (alex.components.nlg.tectotpl.core.node.T method), 76
is_compatible() (alex.components.dm.ontology.Ontology method), 52
is_coord_conj() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
is_coord_taking_1st_pos() (alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowackernagel.MoveCliticsToWackernagel method), 66
is_first_node() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
is_incongruent_numeral() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
is_last_node() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
is_named_entity_label() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
is_personal_role() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 77
is_right_child (alex.components.nlg.tectotpl.core.node.Ordered attribute), 75
is_root (alex.components.nlg.tectotpl.core.node.Node attribute), 74
is_update_server_reachble() (in module alex.utils.config), 104
iteritems() (alex.components.dm.pstate.PDDiscrete method), 53
iteritems() (alex.components.dm.pstate.PDDiscreteOther method), 54

J

Job (class in alex.components.nlg.tectotpl.tool.cluster), 78
jobid (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
JOBNAME_LEGAL_CHARS (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
JuliusASRException, 49
JuliusASRTimeoutException, 49

K

KaldiASRException, 49
KaldiSetupException, 49

L

language_and_selector (alex.components.nlg.tectotpl.core.document.Zone attribute), 72
last_talked_about() (alex.components.dm.ontology.Ontology method), 53
LBP (class in alex.ml.bn.lbp), 92
LBPError, 93
levels (alex.utils.mproc.SystemLogger attribute), 108
lex_anode (alex.components.nlg.tectotpl.core.node.T attribute), 76
Lexicon (class in alex.components.nlg.tectotpl.tool.lexicon.cs), 77
lfu_cache() (in module alex.utils.cache), 102
line_expr (alex.utils.parsers.CamTxtParser attribute), 109
LISTID (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
load() (alex.applications.utils.weather.OpenWeatherMapWeatherFinder method), 47
load() (alex.components.dm.ontology.Ontology method), 53
load() (alex.components.nlg.tectotpl.block.t2a.cs.generatepossessiveadjectiv method), 63
load() (alex.components.nlg.tectotpl.core.block.Block method), 70
load() (alex.components.tts.preprocessing.TTSPreprocessing method), 83
load() (alex.corpustools.grammar_weighted.UniformAlternative method), 89
load() (alex.utils.config.Config method), 103
load_additional_information() (in module alex.applications.PublicTransportInfoCS.data.ontology), 32
load_as_module() (in module alex.utils.config), 104
load_blocks() (alex.components.nlg.tectotpl.core.run.Scenario method), 76
load_compatible_values() (in module alex.applications.PublicTransportInfoCS.data.ontology), 33
load_compatible_values() (in module alex.applications.PublicTransportInfoEN.data.ontology), 43
load_configs() (alex.utils.config.Config class method), 103
load_geo_values() (in module alex.applications.PublicTransportInfoEN.data.ontology), 43
load_includes() (alex.utils.config.Config method), 103
load_list() (in module alex.applications.PublicTransportInfoCS.data.add_cities_to_stops 31
load_list() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
load_list() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.mta_ 39
load_list() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops 40
load_list() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.us_c 40

load_possessive_adj_dict() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 78
load_semantics() (in module alex.corpustools.semscore), 90
load_state_code_dict() (in module alex.applications.PublicTransportInfoEN.data.preprocess_streets_to_ppl), 40
load_street_type_values() (in module alex.applications.PublicTransportInfoEN.data.ontology), 43
load_wavaskey() (in module alex.corpustools.wavaskey), 91
LoadingException, 72
local_lock() (in module alex.utils.mproc), 109
lock (alex.utils.mproc.InstanceID attribute), 108
lock (alex.utils.mproc.SystemLogger attribute), 108
log() (alex.components.hub.calldb.CallDB method), 55
log() (alex.utils.mproc.SystemLogger method), 108
log_and_ipdb_hook() (in module alex.utils.excepthook), 105
log_hook() (in module alex.utils.excepthook), 105
log_info() (in module alex.components.nlg.tectotpl.core.log), 72
log_state() (alex.components.dm.base.DialogueManager method), 50
log_state() (alex.components.dm.base.DialogueState method), 51
log_uri() (alex.components.hub.calldb.CallDB method), 55
log_warn() (in module alex.components.nlg.tectotpl.core.log), 72
logger (alex.utils.excepthook.ExceptionHook attribute), 105
lru_cache() (in module alex.utils.cache), 102

M

main() (in module alex.applications.PublicTransportInfoCS.data.add_cities_to_stops), 31
main() (in module alex.applications.PublicTransportInfoCS.slu.add_to_bootstrap), 35
main() (in module alex.applications.PublicTransportInfoCS.slu.consolidate_keyfiles), 35
main() (in module alex.applications.PublicTransportInfoEN.data.expand_boroughs_script), 41
main() (in module alex.applications.PublicTransportInfoEN.data.expand_cities_script), 41
main() (in module alex.applications.PublicTransportInfoEN.data.expand_states_script), 42
main() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_script), 42
main() (in module alex.applications.PublicTransportInfoEN.data.expand_streets_script), 43
main() (in module alex.applications.PublicTransportInfoEN.data.preprocess
39
main() (in module alex.applications.PublicTransportInfoEN.data.preprocess
39
main() (in module alex.applications.PublicTransportInfoEN.data.preprocess
40
main() (in module alex.applications.PublicTransportInfoEN.data.preprocess
40
main() (in module alex.applications.PublicTransportInfoEN.slu.add_to_boo
44
main() (in module alex.applications.PublicTransportInfoEN.slu.consolidate_
44
main() (in module alex.applications.PublicTransportInfoEN.slu.query_goog
44
main() (in module alex.corpustools.cued2wavaskey), 87
main() (in module alex.corpustools.srilm_ppl_filter), 90
main() (in module alex.tests.test_numpy_with_optimised_ATLAS), 98
map_vehicle() (alex.applications.PublicTransportInfoEN.directions.Googlel method), 45
mark_subpos_tense() (alex.components.nlg.tectotpl.block.t2a.cs.markverbal method), 65
MarkSubject (class in alex.components.nlg.tectotpl.block.t2a.cs.marksubject), 65
MarkVerbalCategories (class in alex.components.nlg.tectotpl.block.t2a.cs.markverbalcategories), 65
matches() (alex.utils.caminfodb.CamInfoDb method), 102
mean() (in module alex.corpustools.num_time_stats), 90
merge() (alex.ml.hypothesis.ConfusionNetwork method), 96
merge() (alex.ml.hypothesis.NBList method), 97
merge() (alex.utils.config.Config method), 103
merge() (in module alex.applications.PublicTransportInfoEN.data.expand_s
42
merge_files() (in module alex.corpustools.merge_uttcns), 89
Message (class in alex.components.hub.messages), 56
meta_slots (alex.components.dm.pstate.PDDiscrete attribute), 53
meta_slots (alex.components.dm.pstate.PDDiscreteOther attribute), 54
min_edit_dist() (in module alex.utils.text), 112
min_edit_ops() (in module alex.utils.text), 112
MODE_TRANSIT (alex.applications.PublicTransportInfoEN.directions.Ro attribute), 46
MODE_WALKING (alex.applications.PublicTransportInfoEN.directions.R attribute), 46
morphcat_case (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_gender (alex.components.nlg.tectotpl.core.node.A attribute), 73

morphcat_grade (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_members (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_mood (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_negation (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_number (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_person (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_pos (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_possgender (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_possnumber (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_subpos (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_tense (alex.components.nlg.tectotpl.core.node.A attribute), 73
morphcat_voice (alex.components.nlg.tectotpl.core.node.A attribute), 73
MoveCliticsToWackernagel (class in alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowackernagel), 65
mph() (alex.components.dm.base.DiscreteValue method), 51
mpv() (alex.components.dm.base.DiscreteValue method), 51
mpvp() (alex.components.dm.base.DiscreteValue method), 51

N

N (class in alex.components.nlg.tectotpl.core.node), 74
name (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
NAME_PREFIX (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
NBList (class in alex.ml.hypothesis), 96
NBListException, 95
NEGATION (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorphcat attribute), 65
nesteddict (class in alex.utils.various), 113
new_aux_node() (alex.components.nlg.tectotpl.block.t2a.addautoword.AddAuxWords method), 67
new_aux_node() (alex.components.nlg.tectotpl.block.t2a.cs.addclausalexpletives method), 60
new_aux_node() (alex.components.nlg.tectotpl.block.t2a.cs.addprepositions.AddPrepositions method), 61
new_aux_node() (alex.components.nlg.tectotpl.block.t2a.cs.addsubconjs.AddSubconjs method), 61
new_dialogue() (alex.components.dm.base.DialogueManager method), 50
NLGException, 81
Node (class in alex.components.nlg.tectotpl.core.node), 74
Node (class in alex.ml.ep.node), 94
NORMAL (alex.utils.text.Escaper attribute), 112
normalise() (alex.components.dm.base.DiscreteValue method), 51
normalise() (alex.ml.ep.node.Node method), 95
normalise() (alex.ml.hypothesis.ConfusionNetwork method), 96
normalise() (alex.ml.hypothesis.NBList method), 97
normalise_path() (in module alex.utils.fs), 107
normalise_text() (in module alex.corpustools.text_norm_cs), 90
normalise_text() (in module alex.corpustools.text_norm_en), 90
normalise_text() (in module alex.corpustools.text_norm_es), 91
normalize() (alex.components.dm.pstate.PDDiscrete method), 53
normalize() (alex.components.dm.pstate.PDDiscreteOther method), 54
ntree (alex.components.nlg.tectotpl.core.document.Zone attribute), 72
NULL (alex.components.dm.pstate.PDDiscrete attribute), 53
NULL (alex.components.dm.pstate.PDDiscreteOther attribute), 54
NUMBER (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorp attribute), 65
number_for() (alex.components.nlg.tectotpl.tool.lexicon.cs.Lexicon method), 78

O

O (class in alex.corpustools.grammar_weighted), 88
OBJECT_STATUS (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
obtain_geo_codes() (alex.applications.PublicTransportInfoEN.time_zone.G method), 47
one() (in module alex.utils), 113
online_update() (in module alex.utils.config), 104
Ontology (class in alex.components.dm.ontology), 52
OntologyException, 53
open_database() (alex.components.hub.calldb.CallDB method), 55
OPEN_PUNCT (alex.components.nlg.tectotpl.block.t2a.cs.capitalizesentsta attribute), 62
OpenWeatherMapWeather (class in alex.applications.utils.weather), 47
OpenWeatherMapWeatherFinder (class in alex.applications.utils.weather), 47

Option (class in alex.corpustools.grammar_weighted), 88
ord (alex.components.nlg.tectotpl.core.document.Bundle attribute), 70
Ordered (class in alex.components.nlg.tectotpl.core.node), 75
OTHER (alex.components.dm.pstate.PDDiscrete attribute), 53
OTHER (alex.components.dm.pstate.PDDiscreteOther attribute), 54

P

P (class in alex.components.nlg.tectotpl.core.node), 75
parent (alex.components.nlg.tectotpl.core.node.Node attribute), 74
parse() (alex.components.slu.templateclassifier.TemplateClassifier method), 82
parse() (alex.corpustools.cuedda.CUEDDialogueAct method), 88
parse() (alex.corpustools.cuedda.CUEDSlot method), 88
parse() (alex.utils.parsers.CamTxtParser method), 109
parse_command() (in module alex.utils.text), 112
parse_line() (alex.components.nlg.tectotpl.block.read.tectotemplates.TectoTemplates method), 57
parse_time() (alex.applications.PublicTransportInfoEN.time_zone.GoogleTimeFinder method), 47
parse_treelet() (alex.components.nlg.tectotpl.block.read.tectotemplates.TectoTemplates method), 58
PDDiscrete (class in alex.components.dm.pstate), 53
PDDiscreteBase (class in alex.components.dm.pstate), 54
PDDiscreteOther (class in alex.components.dm.pstate), 54
persistent_cache() (in module alex.utils.cache), 102
PERSON (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorphcat attribute), 65
PlatformFinderResult (class in alex.applications.PublicTransportInfoCS.platform_info), 38
PlatformInfo (class in alex.applications.PublicTransportInfoCS.platform_info), 38
PlatformInfoTest (class in alex.applications.PublicTransportInfoCS.platform_info_test), 38
postprocess() (alex.components.nlg.tectotpl.block.t2a.addauxwords.AddAuxWords method), 67
postprocess() (alex.components.nlg.tectotpl.block.t2a.cs.addclausalexpletives.AddClausalExpletives method), 60
postprocess() (alex.components.nlg.tectotpl.block.t2a.cs.addprepositions.AddPrepositions method), 61
PowerVAD (class in alex.components.vad.power), 84
preprocess_line() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_script), 42
print_assignment() (in module alex.tools.mturk.bin.mturk), 99
probTable() (alex.ml.ep.node.Goal method), 94
process() (alex.components.tts.preprocessing.TTSPreprocessing method), 83
process_anode() (alex.components.nlg.tectotpl.block.t2a.cs.addcoordpunct.A method), 60
process_anode() (alex.components.nlg.tectotpl.block.t2a.cs.addparentheses. method), 61
process_atree() (alex.components.nlg.tectotpl.block.t2a.cs.addsubordclausep method), 62
process_atree() (alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowac method), 66
process_atree() (alex.components.nlg.tectotpl.block.t2a.cs.vocalizeprepos.V method), 66
process_bundle() (alex.components.nlg.tectotpl.block.util.copytree.CopyTre method), 68
process_bundle() (alex.components.nlg.tectotpl.block.util.eval.Eval method), 69
process_bundle() (alex.components.nlg.tectotpl.block.util.setglobal.SetGlob method), 69
process_bundle() (alex.components.nlg.tectotpl.core.block.Block method), 70
process_clause() (alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowa method), 66
process_document() (alex.components.nlg.tectotpl.block.read.tectotemplate method), 58
process_document() (alex.components.nlg.tectotpl.block.read.yaml.YAML method), 58
process_document() (alex.components.nlg.tectotpl.block.util.eval.Eval method), 69
process_document() (alex.components.nlg.tectotpl.block.write.yaml.YAML method), 69
process_document() (alex.components.nlg.tectotpl.core.block.Block method), 70
process_excepts() (alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr. method), 63
process_excepts() (alex.components.nlg.tectotpl.block.t2a.cs.imposecompla method), 63
process_excepts() (alex.components.nlg.tectotpl.block.t2a.cs.imposepronzag method), 64
process_excepts() (alex.components.nlg.tectotpl.block.t2a.cs.imposerelpron method), 64
process_excepts() (alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpre method), 64
process_excepts() (alex.components.nlg.tectotpl.block.t2a.imposeagreement method), 68
process_places() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
process_tnode() (alex.components.nlg.tectotpl.block.t2a.addauxwords.AddA method), 67
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.addappositionpu method), 58

process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundfuture.AddAuxVerbCompoundFuture method), 58
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpassive.AddAuxVerbCompoundPassive method), 59
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.addauxverbcompoundpast.AddAuxVerbCompoundPast method), 59
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.addauxverbconditional.AddAuxVerbConditional method), 59
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.addauxverbmodal.AddAuxVerbModal method), 59
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.addreflexiveparticles.AddReflexiveParticles method), 61
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.deletesuperfluousauxs.DeleteSuperfluousAuxs method), 62
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.dropsubjpersprons.DropSubjPersProns method), 62
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.generatepossessiveadjectives.GeneratePossessiveAdjectives method), 63
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorphcat method), 65
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.markverbalcategories.MarkVerbalCategories method), 65
process_tnode() (alex.components.nlg.tectotpl.block.t2a.cs.projectclausenumber.ProjectClauseNumber method), 66
process_tnode() (alex.components.nlg.tectotpl.block.t2a.imposeagreement.ImposeAgreement method), 68
process_ttree() (alex.components.nlg.tectotpl.block.t2a.cs.addsentfinalpunct.AddSentFinalPunct method), 61
process_ttree() (alex.components.nlg.tectotpl.block.t2a.cs.marksubject.MarkSubject method), 65
process_ttree() (alex.components.nlg.tectotpl.block.t2a.cs.reversenumbernoundependency.ReverseNumberNounDependency method), 66
process_zone() (alex.components.nlg.tectotpl.block.a2w.cs.concatenatetokens.ConcatenateTokens method), 57
process_zone() (alex.components.nlg.tectotpl.block.a2w.cs.removerepeatedtokens.RemoveRepeatedTokens method), 57
process_zone() (alex.components.nlg.tectotpl.block.t2a.copyttree.CopyTTree method), 67
process_zone() (alex.components.nlg.tectotpl.block.t2a.cs.capitalizesentstart.CapitalizeSentStart method), 62
process_zone() (alex.components.nlg.tectotpl.block.util.eval.Eval method), 69
process_zone() (alex.components.nlg.tectotpl.core.block.Block method), 70
ProjectClauseNumber (class in alex.components.nlg.tectotpl.block.t2a.cs.projectclausenumber), 66
PRONOUNS (alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr.ImposePronZAgr attribute), 64
prune() (alex.components.dm.base.DiscreteValue method), 51
prune() (alex.ml.hypothesis.ConfusionNetwork method), 96
PTICSHDCPolicyException, 37
PTIENHDCPolicyException, 46
ptree (alex.components.nlg.tectotpl.core.document.Zone attribute), 78

Q

QSUB_MEMORY_CMD (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
QSUB_MULTICORE_CMD (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79

R

random() (in module alex.applications.PublicTransportInfoCS.data.get_citie 32
random() (in module alex.ml.ep.test), 95
Rdb (class in alex.utils.rdb), 110
re_literal() (alex.utils.text.Escaper static method), 112
re_literal_list() (alex.utils.text.Escaper static method), 112
read_compatibility() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
read_database() (alex.components.hub.calldb.CallDB method), 55
read_expansions() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
read_exports() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
read_first_column() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 42
read_prev_compatibility() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.comp 39
read_two_columns() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip 43
readRules() (alex.components.slu.templateclassifier.TemplateClassifier method), 82
ref_attrib (alex.components.nlg.tectotpl.core.node.A attribute), 73
ref_attrib (alex.components.nlg.tectotpl.core.node.EffectiveRelations attribute), 73
ref_attrib (alex.components.nlg.tectotpl.core.node.InClause attribute), 73
ref_attrib (alex.components.nlg.tectotpl.core.node.N attribute), 74
ref_attrib (alex.components.nlg.tectotpl.core.node.Node attribute), 74
ref_attrib (alex.components.nlg.tectotpl.core.node.Ordered attribute), 75

ref_attrib (alex.components.nlg.tectotpl.core.node.P attribute), 75
ref_attrib (alex.components.nlg.tectotpl.core.node.T attribute), 76
REG (in module alex.applications.PublicTransportInfoCS.crws_enums), 36
release() (alex.utils.filelock.FileLock method), 106
release_database() (alex.components.hub.calldb.CallDB method), 55
REMMASK (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
remove() (alex.components.dm.pstate.PDDiscreteBase method), 54
remove() (alex.components.nlg.tectotpl.core.node.Node method), 74
remove() (alex.ml.hypothesis.ConfusionNetwork method), 96
remove_aux_anodes() (alex.components.nlg.tectotpl.core.node.T method), 76
remove_backref() (alex.components.nlg.tectotpl.core.document.Document method), 71
remove_dependency() (alex.components.nlg.tectotpl.tool.cluster.Job method), 79
remove_duplicities() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv), 39
remove_duplicities() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment), 40
remove_duplicities() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv), 40
remove_dups_stable() (in module alex.utils.various), 113
remove_following_duplicities() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.mta_to_csv), 39
remove_following_duplicities() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops_to_streets_experiment), 40
remove_following_duplicities() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.us_cities_to_csv), 40
remove_listener() (alex.utils.fs.GrepFilter method), 107
remove_node() (alex.components.nlg.tectotpl.core.document.Document method), 71
remove_reference() (alex.components.nlg.tectotpl.core.node.Node method), 74
remove_spaces() (in module alex.corpustools.grammar_weighted), 89
RemoveRepeatedTokens (class in alex.components.nlg.tectotpl.block.a2w.cs.removerepeatedtokens), 57
report (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
reset_morphcat() (alex.components.nlg.tectotpl.core.node.A method), 73
reset_on_change() (alex.components.dm.ontology.Ontology method), 53
resolve_imperative() (alex.components.nlg.tectotpl.block.t2a.cs.markverbal method), 65
resolve_infinitive() (alex.components.nlg.tectotpl.block.t2a.cs.markverbalca method), 65
restart() (alex.components.dm.base.DialogueState method), 51
ReverseNumberNounDependency (class in alex.components.nlg.tectotpl.block.t2a.cs.reversenumbernoundep 66
root (alex.components.nlg.tectotpl.core.node.Node attribute), 74
root() (in module alex.utils.env), 105
Route (class in alex.applications.PublicTransportInfoEN.directions), 45
ROUTE_FLAGS (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
RouteLeg (class in alex.applications.PublicTransportInfoEN.directions), 45
RouteStep (class in alex.applications.PublicTransportInfoEN.directions), 46
Rule (class in alex.corpustools.grammar_weighted), 88
run() (alex.ml.bn.lbp.BP method), 92
run() (alex.ml.bn.lbp_ext.BP method), 93
run() (alex.utils.fs.GrepFilter method), 107
run() (alex.utils.sessionlogger.SessionLogger method), 110
RuntimeException, 72

S

S (class in alex.corpustools.grammar_weighted), 88
sample() (alex.corpustools.grammar_weighted.Alternative method), 88
sample() (alex.corpustools.grammar_weighted.GrammarGen method), 88
sample() (alex.corpustools.grammar_weighted.Option method), 88
sample() (alex.corpustools.grammar_weighted.Sequence method), 88
sample() (alex.corpustools.grammar_weighted.Terminal method), 89
sample() (alex.corpustools.grammar_weighted.UniformAlternative method), 89
sample_uniq() (alex.corpustools.grammar_weighted.GrammarGen method), 88
save_database() (in module alex.corpustools.ufaldatabase), 91
save_list() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_scrip

43
save_out() (in module alex.applications.PublicTransportInfoEN.data.expand_stops_script), 43
save_set() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.compatibility_sd.plot_manual), 39
save_wavaskey() (in module alex.corpustools.wavaskey), 91
Scenario (class in alex.components.nlg.tectotpl.core.run), 76
ScenarioException, 72
score() (in module alex.corpustools.semscore), 90
score_da() (in module alex.corpustools.semscore), 90
score_file() (in module alex.corpustools.semscore), 90
script_path() (in module alex.utils), 113
sd() (in module alex.corpustools.num_time_stats), 90
SEARCHMODE (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
SemHubException, 48
Sequence (class in alex.corpustools.grammar_weighted), 88
serialize_bundle() (alex.components.nlg.tectotpl.block.write.yaml.YAML method), 69
serialize_node() (alex.components.nlg.tectotpl.block.write.yaml.YAML method), 69
serialize_tree() (alex.components.nlg.tectotpl.block.write.yaml.YAML method), 70
serialize_zone() (alex.components.nlg.tectotpl.block.write.yaml.YAML method), 70
session_end() (alex.utils.mproc.SystemLogger method), 108
session_start() (alex.utils.mproc.SystemLogger method), 108
session_system_log() (alex.utils.mproc.SystemLogger method), 108
SessionClosedException, 105
SessionLogger (class in alex.utils.sessionlogger), 110
SessionLoggerException, 106
set_and_ret() (in module alex.corpustools.num_time_stats), 90
set_attr() (alex.components.nlg.tectotpl.core.node.Node method), 75
set_case() (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorphcat method), 65
set_cfg() (alex.utils.sessionlogger.SessionLogger method), 110
set_close_event() (alex.utils.sessionlogger.SessionLogger method), 110
set_deref_attr() (alex.components.nlg.tectotpl.core.node.Node method), 75
set_hook() (alex.utils.excepthook.ExceptionHook class method), 105
set_online_update_server() (in module alex.utils.config), 104
set_persitent_cache_content() (in module alex.utils.cache), 102
set_perspron_categories() (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorph method), 65
set_proc_name() (in module alex.utils.procname), 109
SetGlobal (class in alex.components.nlg.tectotpl.block.util.setglobal), 69
setParents() (alex.ml.ep.node.Goal method), 94
setUp() (alex.utils.test_fs.TestFind method), 110
setValues() (alex.ml.ep.node.Goal method), 94
setValues() (alex.ml.ep.node.GroupingGoal method), 94
shift_after_node() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
shift_after_subtree() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
shift_before_node() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
shift_before_subtree() (alex.components.nlg.tectotpl.core.node.Ordered method), 75
should_agree() (alex.components.nlg.tectotpl.block.t2a.cs.imposeattragr.Im method), 63
should_agree() (alex.components.nlg.tectotpl.block.t2a.cs.imposecomplagr. method), 63
should_agree() (alex.components.nlg.tectotpl.block.t2a.cs.imposepronzagr.I method), 64
should_agree() (alex.components.nlg.tectotpl.block.t2a.cs.imposerelpronagr method), 64
should_agree() (alex.components.nlg.tectotpl.block.t2a.cs.imposesubjpreda method), 64
should_agree() (alex.components.nlg.tectotpl.block.t2a.imposeagreement.In method), 68
should_ignore() (alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowa method), 66
SimpleUpdater (class in alex.components.dm.pstate), 54
slot_has_value() (alex.components.dm.ontology.Ontology method), 53
slot_is_binary() (alex.components.dm.ontology.Ontology method), 53
slots_system_confirms() (alex.components.dm.ontology.Ontology method), 53
slots_system_requests() (alex.components.dm.ontology.Ontology method), 53
slots_system_selects() (alex.components.dm.ontology.Ontology method), 53
SLUConfigurationException, 82
SLUException, 82
SLUHyp (class in alex.components.hub.messages), 56
sort() (alex.ml.hypothesis.ConfusionNetwork method), 96
space_size (alex.components.dm.pstate.PDDiscreteOther attribute), 54
spell_if_number() (in module

alex.applications.PublicTransportInfoEN.site_preprocessing), 46
split_by() (in module alex.utils.text), 112
split_by_comma() (in module alex.utils.text), 113
split_to_bins() (in module alex.utils.various), 113
splitOff() (alex.ml.ep.node.GroupingNode method), 94
srilm_scores() (in module alex.corpustools.srilm_ppl_filter), 90
ST (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
state (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
State (class in alex.components.dm.state), 54
state_class (alex.components.dm.tracker.StateTracker attribute), 54
StateTracker (class in alex.components.dm.tracker), 54
station_name_splitter (alex.applications.PublicTransportInfoCS.platform_info attribute), 38
stick_place_in_front() (in module alex.applications.PublicTransportInfoEN.data.preprocessing), 39
submit() (alex.components.nlg.tectotpl.tool.cluster.Job method), 80
SVCSTATE (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
synthesize() (alex.components.tts.base.TTSInterface method), 83
SystemLogger (class in alex.utils.mproc), 108

T

T (class in alex.components.nlg.tectotpl.core.node), 75
T (class in alex.corpustools.grammar_weighted), 88
tearDown() (alex.utils.test_fs.TestFind method), 111
TectoTemplates (class in alex.components.nlg.tectotpl.block.read.tectotemplates), 57
TemplateClassifier (class in alex.components.slu.templateclassifier), 82
TemplateNLGException, 81
Terminal (class in alex.corpustools.grammar_weighted), 88
test_cycles() (alex.utils.test_fs.TestFind method), 111
test_depth() (alex.utils.test_fs.TestFind method), 111
test_globs() (alex.utils.test_fs.TestFind method), 111
test_grep_filter() (in module alex.utils.fs), 107
test_ignore_globs() (alex.utils.test_fs.TestFind method), 111
test_iter() (alex.ml.test_hypothesis.TestConfusionNetwork method), 97
test_matching() (alex.applications.PublicTransportInfoCS.platform_info_test.PlatformInfoTest method), 38
test_parse_command() (alex.utils.test_text.TestString method), 111
test_resume() (alex.ml.test_hypothesis.TestConfusionNetwork method), 97
test_split_by() (alex.utils.test_text.TestString method), 111
test_symlinks1() (alex.utils.test_fs.TestFind method), 111
test_wrong_args() (alex.utils.test_fs.TestFind method), 111
TestConfusionNetwork (class in alex.ml.test_hypothesis), 97
TestFind (class in alex.utils.test_fs), 110
TestString (class in alex.utils.test_text), 111
TextHubException, 48
Time (class in alex.applications.PublicTransportInfoEN.time_zone), 47
TIME_POLL_DELAY (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
TIME_QUERY_DELAY (alex.components.nlg.tectotpl.tool.cluster.Job attribute), 79
TIMETABLE_FLAGS (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
tmphs() (alex.components.dm.base.DiscreteValue method), 51
tmpvs() (alex.components.dm.base.DiscreteValue method), 51
tmpvsp() (alex.components.dm.base.DiscreteValue method), 51
to_project_path() (in module alex.utils.config), 104
track_confirmed_call() (alex.components.hub.calldb.CallDB method), 55
track_disconnected_call() (alex.components.hub.calldb.CallDB method), 55
transcription (alex.corpustools.cued2utt_da_pairs.TurnRecord attribute), 86
Travel (class in alex.applications.PublicTransportInfoEN.directions), 46
TRCAT (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
TreexException, 72
TRSUBCAT (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
TTDETAILS (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
TTERR (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
TTGP (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
TTINFODETAILS (in module alex.applications.PublicTransportInfoCS.crws_enums), 37

TTLANG (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
ttree (alex.components.nlg.tectotpl.core.document.Zone attribute), 72
TTSException, 83
TTSInterface (class in alex.components.tts.base), 83
TTSPreprocessing (class in alex.components.tts.preprocessing), 83
TTSPreprocessingException (class in alex.components.tts.preprocessing), 83
TTSText (class in alex.components.hub.messages), 56
Turn (class in alex.ml.ep.turn), 95
TurnRecord (class in alex.corpustools.cued2utt_da_pairs), 86

U

UA (class in alex.corpustools.grammar_weighted), 89
unescape() (alex.utils.text.Escaper method), 112
unfold_lists() (alex.utils.config.Config method), 103
UniformAlternative (class in alex.corpustools.grammar_weighted), 89
update() (alex.components.dm.base.DialogueState method), 51
update() (alex.components.dm.pstate.PDDiscrete method), 53
update() (alex.components.dm.pstate.PDDiscreteOther method), 54
update() (alex.components.dm.pstate.SimpleUpdater method), 54
update() (alex.components.dm.state.State method), 54
update() (alex.ml.ep.node.ConstChangeGoal method), 94
update() (alex.ml.ep.node.Goal method), 94
update() (alex.ml.ep.node.GroupingGoal method), 94
update() (alex.utils.config.Config method), 103
update_prob() (alex.ml.hypothesis.ConfusionNetwork method), 96
update_slot() (alex.components.dm.pstate.SimpleUpdater method), 54
update_state() (alex.components.dm.tracker.StateTracker method), 54

V

valid_args (alex.components.nlg.tectotpl.block.util.eval.Eval attribute), 69
var() (in module alex.corpustools.num_time_stats), 90
VEHICLE_TYPE_MAPPING (alex.applications.PublicTransportInfoEN.directions.GoogleRouteLegStep attribute), 45
verb_group_root() (alex.components.nlg.tectotpl.block.t2a.cs.movecliticstowackernagel.MoveCliticsToWackernagel method), 66
VF (in module alex.applications.PublicTransportInfoCS.crws_enums), 37
vocalize() (alex.components.nlg.tectotpl.block.t2a.cs.vocalizeprepos.VocalizePrepos method), 67
VocalizePrepos (class in alex.components.nlg.tectotpl.block.t2a.cs.vocalizeprepos), 66
VOICE (alex.components.nlg.tectotpl.block.t2a.cs.initmorphcat.InitMorphc attribute), 65
VoipHubException, 48
VoipIOException, 55

W

wait() (alex.components.nlg.tectotpl.tool.cluster.Job method), 80
walk() (alex.utils.various.nesteddict method), 113
warning() (alex.utils.mproc.SystemLogger method), 108
Weather (class in alex.applications.utils.weather), 47
WeatherFinder (class in alex.applications.utils.weather), 47
WeatherPoint (class in alex.applications.utils.weather), 47
word_for_number() (in module alex.components.nlg.tools.en), 80
write() (alex.utils.fs.GrepFilter method), 107
write_asrhyp_sem() (in module alex.corpustools.cued2utt_da_pairs), 87
write_asrhyp_semhyp() (in module alex.corpustools.cued2utt_da_pairs), 87
write_data() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.mta_ 39
write_data() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.stops 40
write_data() (in module alex.applications.PublicTransportInfoEN.data.preprocessing.us_c 40
write_data() (in module alex.corpustools.cued2utt_da_pairs), 87
write_readline() (alex.components.hub.hub.Hub method), 55
write_trns_sem() (in module alex.corpustools.cued2utt_da_pairs), 87

Y

YAML (class in alex.components.nlg.tectotpl.block.read.yaml), 58
YAML (class in alex.components.nlg.tectotpl.block.write.yaml), 69

Z

zone (alex.components.nlg.tectotpl.core.node.Node attribute), 75
Zone (class in alex.components.nlg.tectotpl.core.document), 71