D5.2 – MARIO Reading and Listening Component

Project Acronym:	**MARIO**
Project Title:	**Managing active and healthy aging with use of caring service robots**
Project Number:	**643808**
Call:	**H2020-PHC-2014-single-stage**
Topic:	**PHC-19-2014**
Type of Action:	**RIA**

Managing Active & Healthy Ageing with Service Robots

D5.2 – MARIO Reading and Listening Component	
Work Package	WP5
Due Date	M22
Submission Date	30/11/2016
Start Date of the Project	01/02/2015
Duration of the Project	36 Months
Organisation Responsible of Deliverable	CNR
Version	1.0
Status	Intermediate
Author name(s)	Valentina Presutti (CNR) Alessandro Russo (CNR) Luigi Asprino (CNR) Aldo Gangemi (CNR) Domenico Pisanelli (CNR) Massimiliano Raciti (R2M) Diego Reforgiato Recupero (R2M) André Freitas (PASSAU) Lazaros Penteridis (Ortelio) Alex Gkiokas (Ortelio)
Nature	☐ R – Report ☐ P – Prototype ☐ D – Demonstrator ☒ O – Other
Dissemination Level	☒ PU – Public ☐ CO – Confidential, only for members of the Consortium (including the Commission) ☐ RE – Restricted to a group specified by the Consortium (including the Commission Services)

Revision History			
Version	Date	Modified by	Comments
0.1	5/11/2016	Valentina Presutti	First Draft and Document Structure
0.2	5/12/2016	Valentina Presutti	Draft Review
1.0	15/12/2016	Alessandro Russo and Valentina Presutti	Document revision according to Diego Reforgiato's review
2.0	25/1/2017	Luigi Asprino	• Added description of the Ontology Matching (Slides 51-56) • Added description of the paraphrase dataset (Slides 46-47) • Added description of the Human activities dataset and description of method for populating the KB with Object-relation relations (Slides 57-63)

Outline and Contents

Overview of the report content

Brief introduction on the notion of frames

Main software components developed in Task 5.2

- Vocal interface
- Semantic Correction component
- Natural language understanding component (Ludwig)

Overview

This report briefly describes a set of software components that are under development within WP5 and that contribute to the MARIO software system architecture. It is intended to provide the reviewers with an intermediate update on Task 5.2.

Most of the software components' code is available in the MARO project svn repository https://mare.istc.cnr.it/svn/mario/; some of them, where specified, are kept in other repositories and made available as REST services (and possibly associated with an online demo)

Some of the software components are off-the-shelf product, others implement state-of-the-art techniques, other implement novel approaches. Some of the latter are associated with articles that have been published or accepted to be published in scientific venues. We include references to these works in the section "*Publications describing research in this task*" *(slide n.50).* Please, consider these papers as integrating part of this deliverable.

Task 5.2

This task deals with MARIO's capability to understand natural language requests from a patient with dementia (PwD)

It relies on the results of WP5 Task 5.1 (see deliverable D5.1) as far as the background knowledge and knowledge models that it uses are concerned

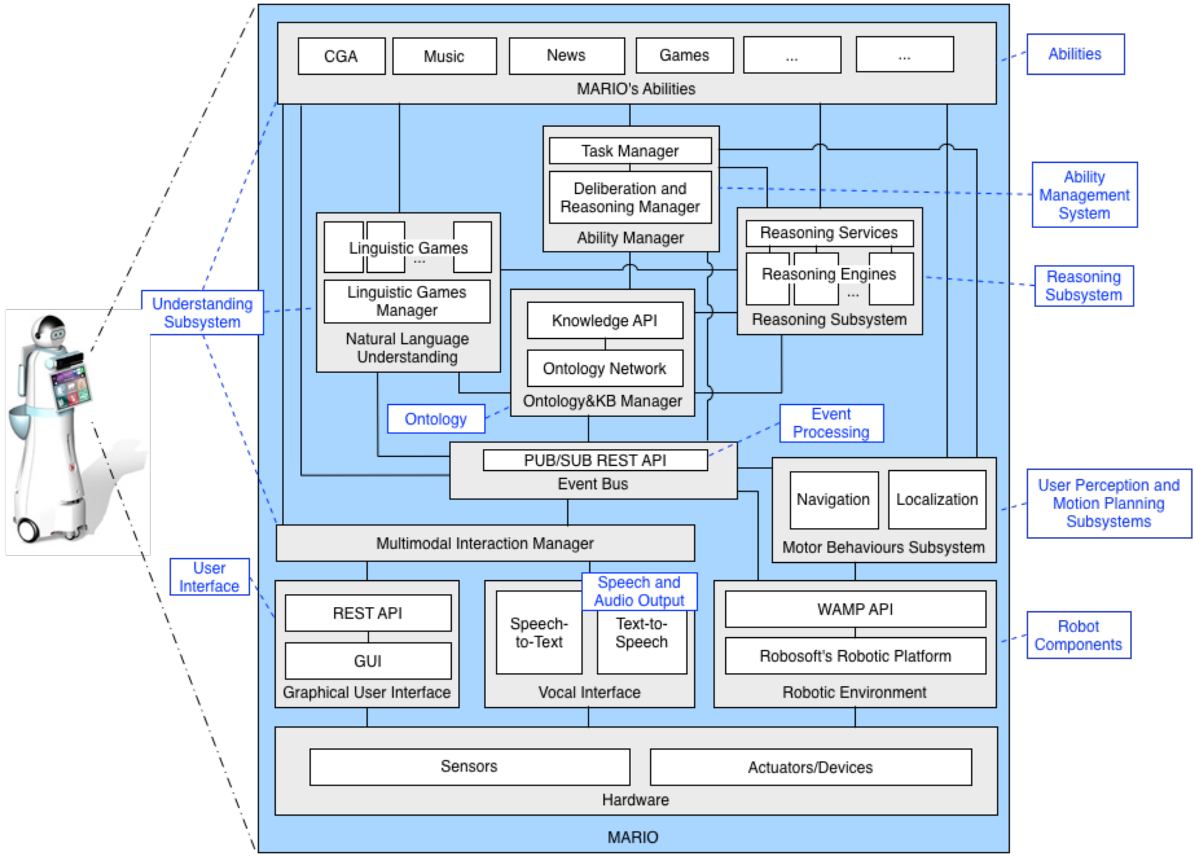
It provides support to tasks in WP4 and WP6 as far as natural-language based interaction is concerned.

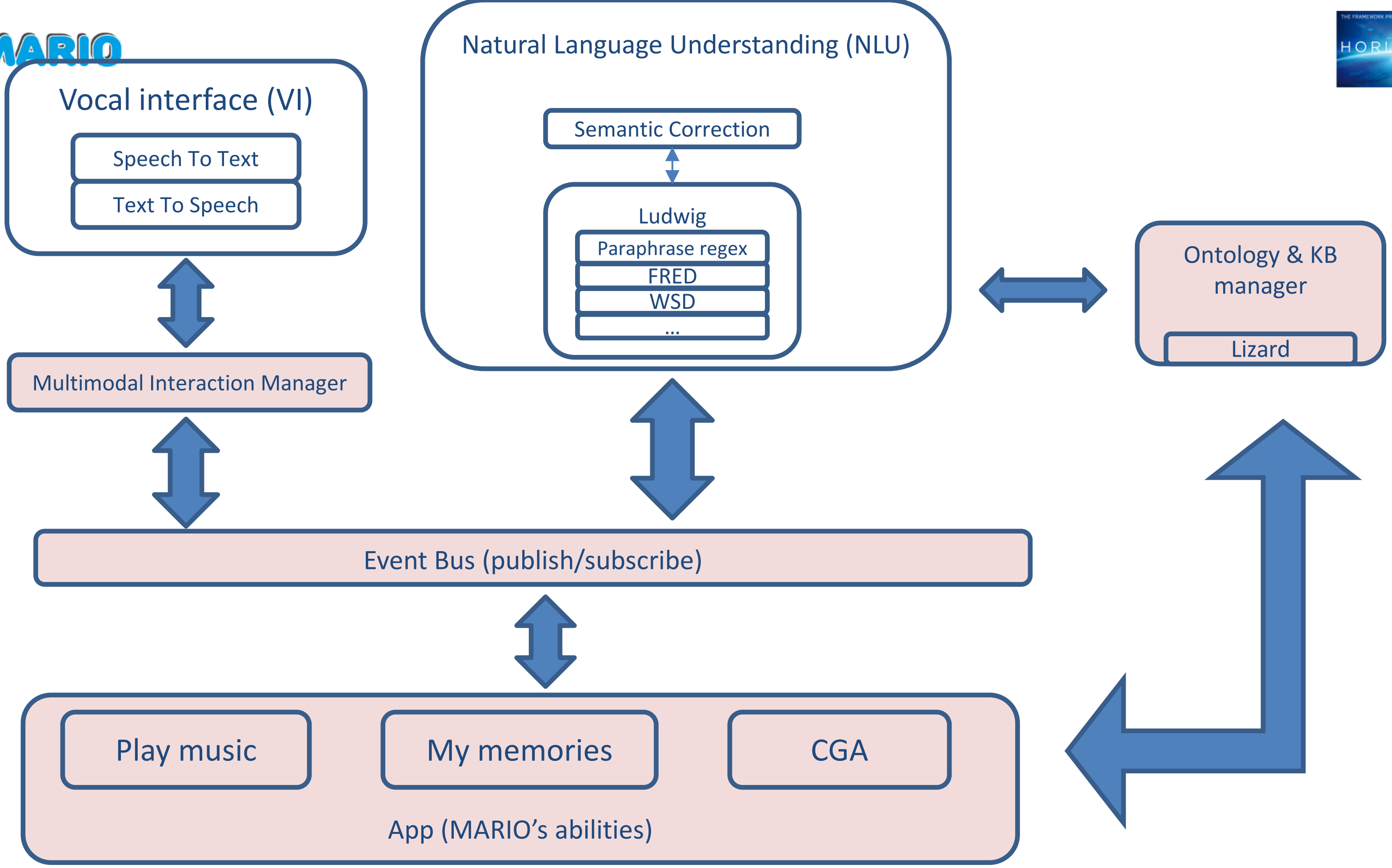
Referring to MARIO architecture, this task contributes to the following components:

- Vocal interface
- Natural Language Understanding

Most of the software components interact with the rest of the system through the Event Bus component, which is managed by the Ability Manager (also called *Marvin*)

To this aim, components subscribe and/or publish to "topics" (see Deliverable 6.2 for details about Marvin)





- This picture highlights the most relevant dependencies and the flow among the components developed in this task
- The NLU includes a set of subcomponents for processing natural language text (see next slides). It communicates with the VI by means of the Event Bus, which provides a publish/subscribe mechanism
- NLU directly interacts with MARIO Knowledge Base via *Lizard* REST API (see Deliverable 5.1) for retrieving and storing data
- Pink components are developed in other project's tasks (T5.1, WP6 and WP4)

Cognitive Frames

An important aspect of MARIO NLU component is that it relies on the concept of cognitive frames [1,2]

A frame is a data structure that represents a stereotyped situation [2]

It is formalised as a n-ary relation where its arguments include an event described by the frame and its participating entities [3]

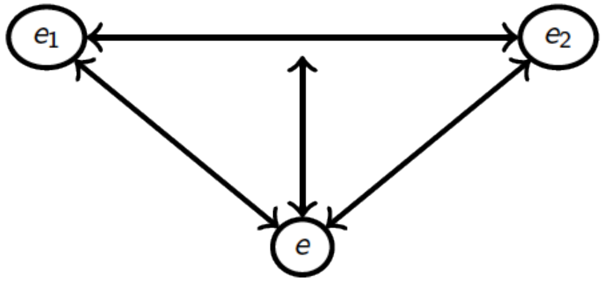
Representing frames

N-ary relation $f(e, e_1,…e_n)$

f is a first order logic relation

e is a variable for any event or situation described by *f*

e_i is a variable for any of the entity arguments of *f*



A semantic web (OWL) n-ary relation pattern

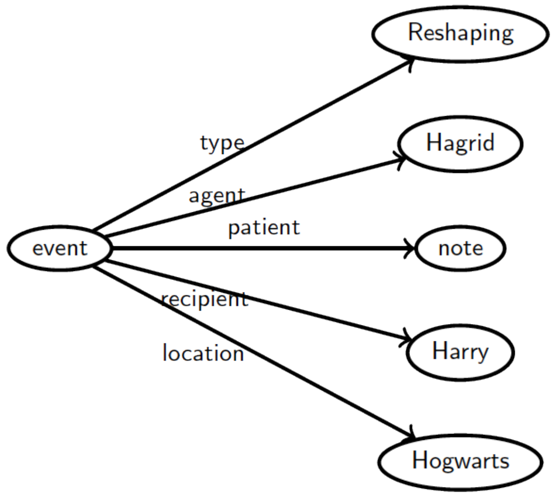
the n-ary relation is the reification of *f*, i.e. *e*

the n objects represent the arguments of *f*

the n argument relations are binary projections of *f* including *e*

co-participation relations are binary projections of *f* not including *e*

"Hagrid rolled up a note for Harry in Hogwarts"



Frames

For example the linguistic frame “Cure” defines a context, a *cure* in this case, based on which we can interpret the terms that refer to things that participate in it, such as a patient, a medication, and a healer

For example the sentence “Doctors alleviated his suffering” is an occurrence of this frame in natural language text

Next slide depicts how this frame is codified within the FrameNet repository of linguistic frames [4] (included in Framester, cf. Deliverable 5.1)

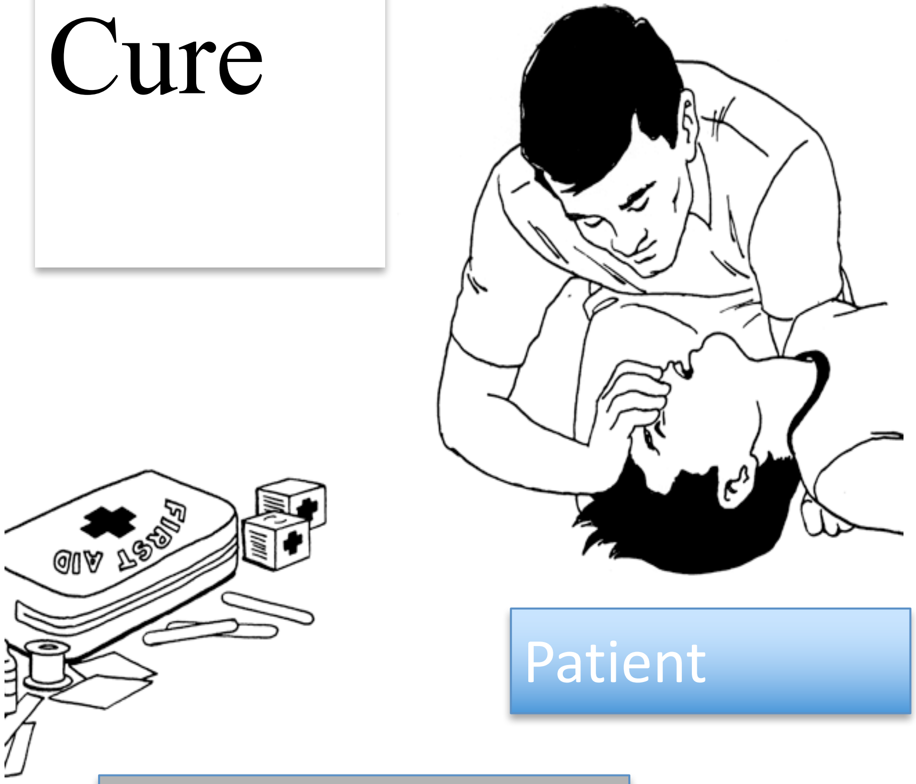
FrameNet frame "Cure"

This frame deals with a Healer treating and curing an Affliction (the injuries, disease, or pain) of the Patient, sometimes also mentioning the use of a particular Treatment or Medication.

Medication [Med] **Semantic Type** Physical_entity	The injested, applied, injected, etc. substance designed to cure the Patient. He needs prolonged TREATMENT with anibiotics. Note the tight relationship between Treatment and Medicine.
Healer [Hlr] **Semantic Type** Sentient	The Healer, anyone who treats or cures the Patient, occurs as the External Argument of verbs: Doctors ALLEVIATED his suffering.

Cure

Healer



Patient

Medication

http://framenet.icsi.berkeley.edu/

Vocal interface

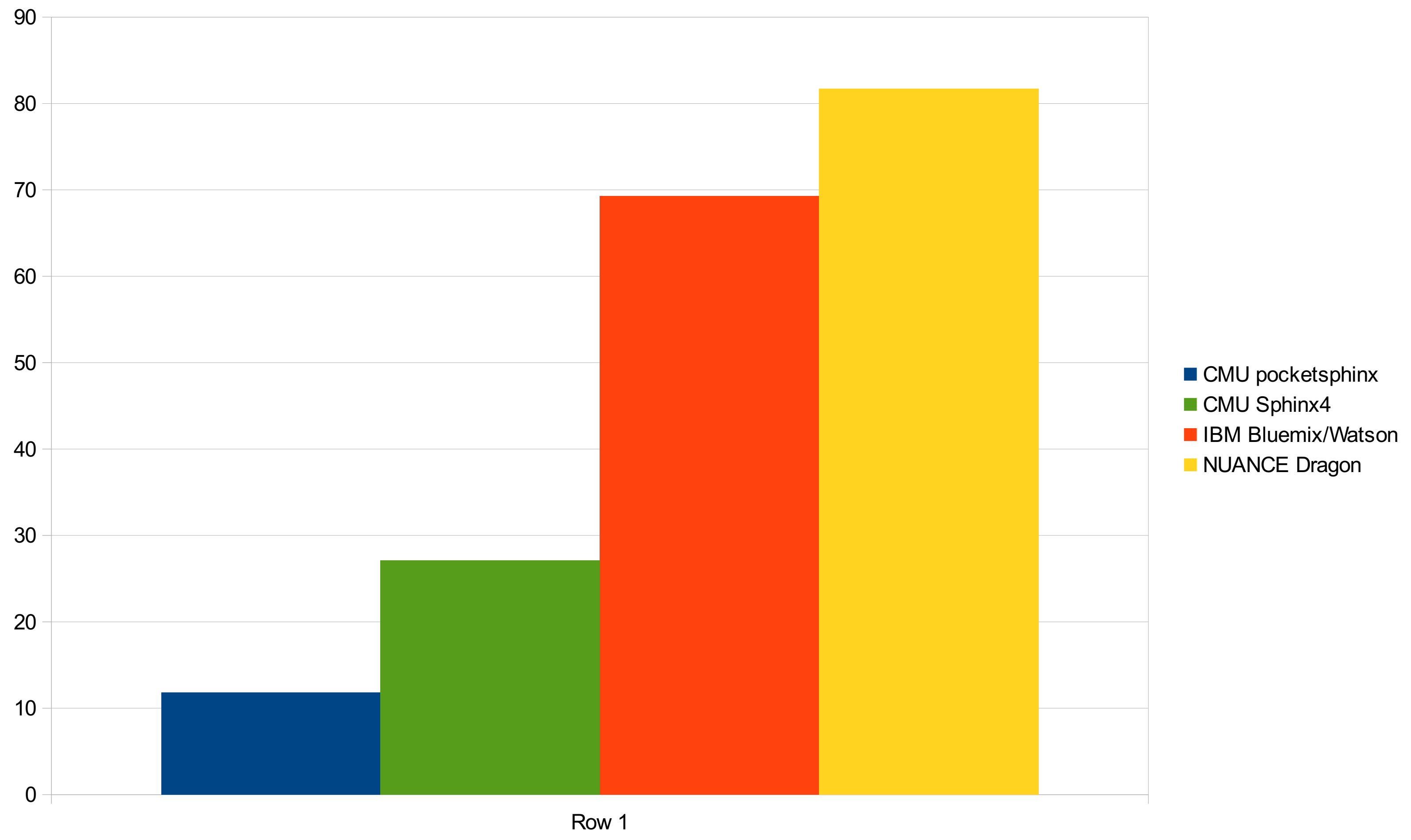
Identifying the best Speech To Text

We have conducted a Survey for selecting the most appropriate and possibly best performing component providing Speech to Text technology

We used an open dictionary extracted from data provided by the partners IRCCS and Galway from anonymised patient dialogs

The next two slides summarise the result of this Survey

F1 score for open dictionary of the Speech recognition API/SDK and engines tested



Other important parameters taken into account

CMU pocketshpinx and CMU Sphinx4 are native C and Java applications, respectively; whereas IBM Bluemix/Watson, Nuance Dragon, Speechmatics, Project Oxford, Google Speech Recognition API are cloud-based, hence they require Internet connection and there is no control over the data being transmitted and factors such as network latency.

The only APIs that support Italian are Dragon and Google's.

Other important parameters taken into account

Important cons that led to rejection of the respective systems:

- **Speechmatics** was extremely slow and very inaccurate
- **Project Oxford** only supports .Net and would require workarounds in order to be used from Linux
- **IBM Bluemix/Watson** does not support Italian and is less accurate than Dragon
- **CMU pocketsphinx** and **CMU Sphinx4** had low accuracy for open dictionary
- **Google's API** is undocumented

Dragon results to be the best choice for MARIO’s requirements

Speech2Text

Nuance **Dragon Naturally Speaking** (DNS) 13 premium

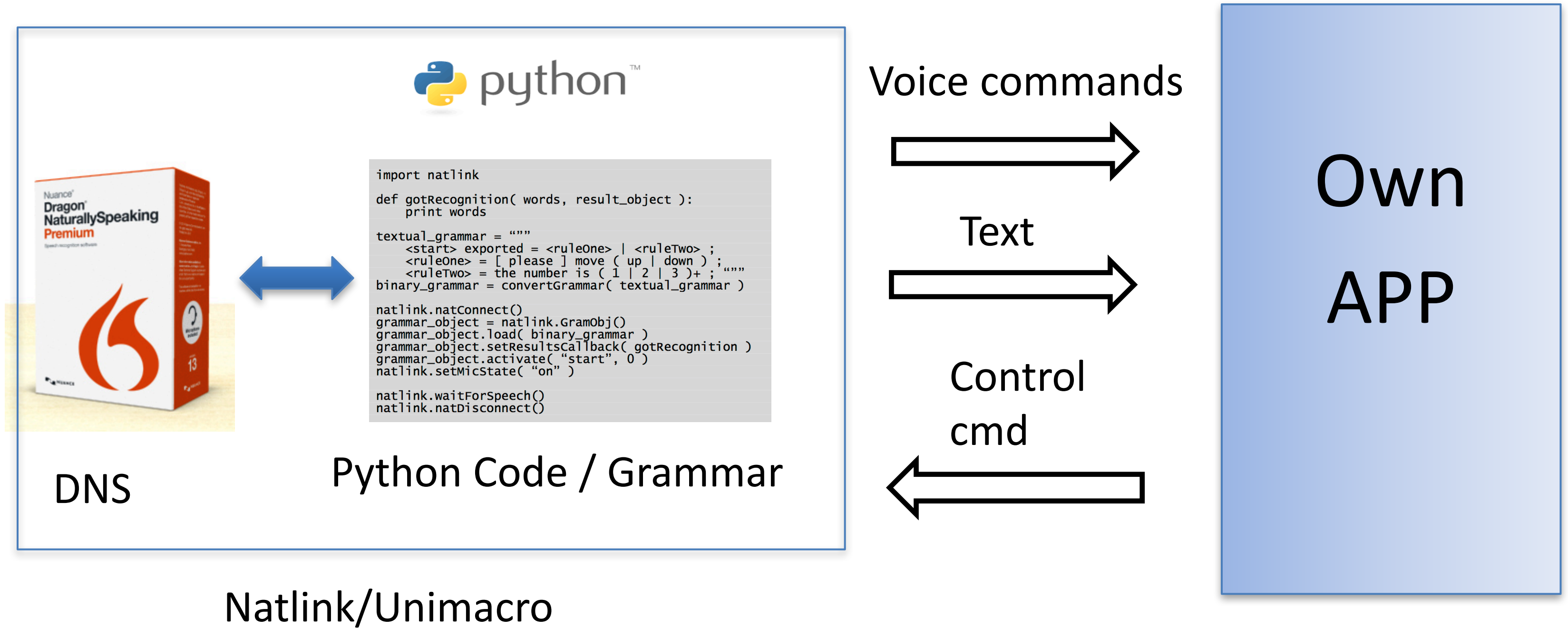
- Pros
  - Best off-the-shelf S2T software
  - Italian and English are fully supported
  - Runs locally – no internet connection required
  - Runs on Windows – can directly use the Kinect input
- Cons
  - Not really designed to develop applications with it
  - Scripting language exists (in Visual Basic/C#) but just to create new voice commands
  - Requires voice training for each user to improve recognition accuracy

Solution: DNS 13 + wrapper code to interact with it (Natlink/Unimacro)

We bought license rights that cover up to three installations. Additional licenses will be acquired once the other platforms will be delivered

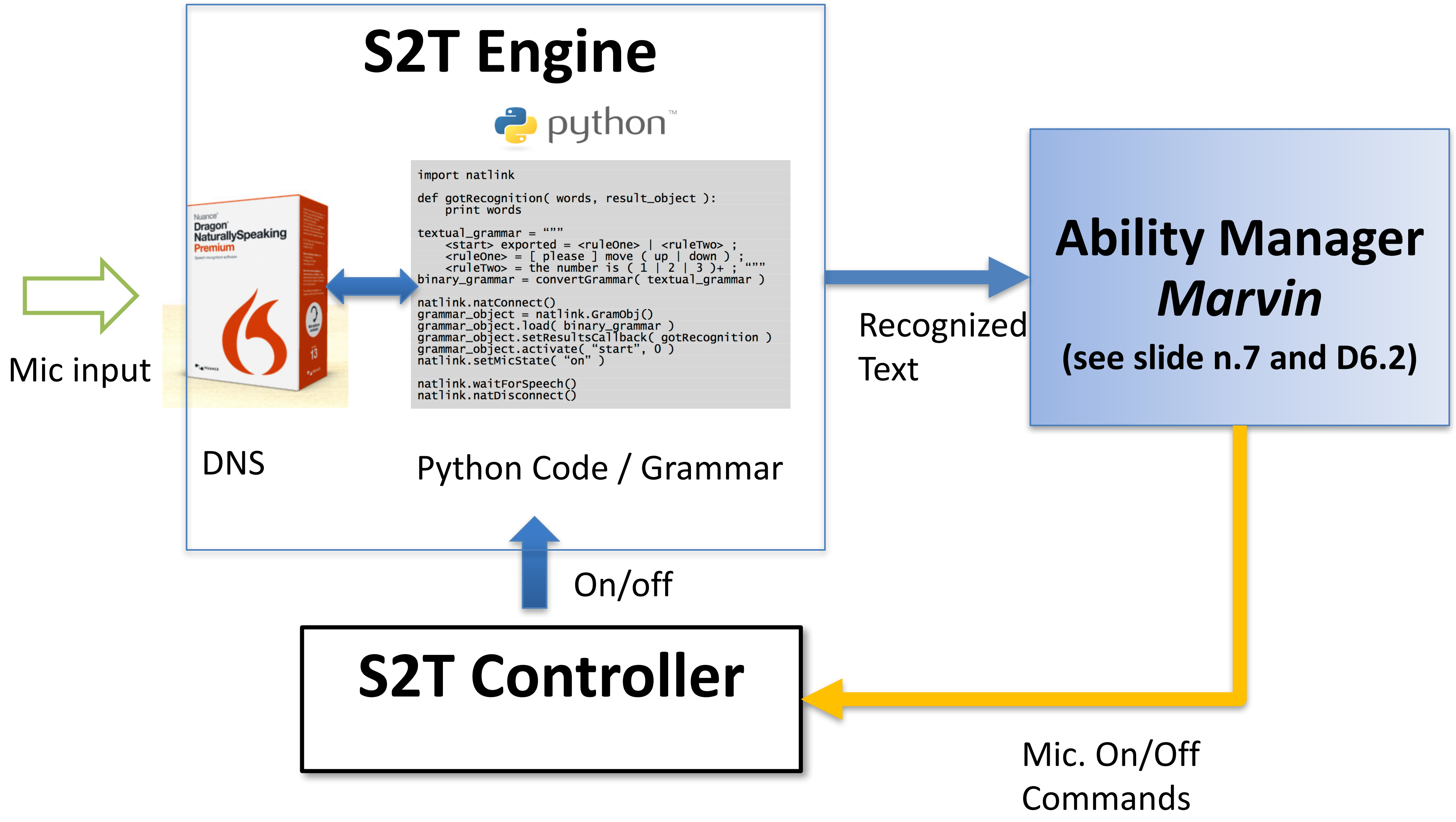
Natlink/Unimacro

- **Natlink/Unimacro**: an extension to Dragon NaturallySpeaking that allows us to develop command and control (speech) macros using the Python programming language http://qh.antenna.nl/unimacro/index.html



```
import natlink

def gotRecognition( words, result_object ):
    print words

textual_grammar = “””
    <start> exported = <ruleOne> | <ruleTwo> ;
    <ruleOne> = [ please ] move ( up | down ) ;
    <ruleTwo> = the number is ( 1 | 2 | 3 )+ ; “””
binary_grammar = convertGrammar( textual_grammar )

natlink.natConnect()
grammar_object = natlink.GramObj()
grammar_object.load( binary_grammar )
grammar_object.setResultsCallback( gotRecognition )
grammar_object.activate( “start”, 0 )
natlink.setMicState( “on” )

natlink.waitForSpeech()
natlink.natDisconnect()
```

Speech2Text Architecture



```
import natlink

def gotRecognition( words, result_object ):
    print words

textual_grammar = “””
    <start> exported = <ruleOne> | <ruleTwo> ;
    <ruleOne> = [ please ] move ( up | down ) ;
    <ruleTwo> = the number is ( 1 | 2 | 3 )+ ; “””
binary_grammar = convertGrammar( textual_grammar )

natlink.natConnect()
grammar_object = natlink.GramObj()
grammar_object.load( binary_grammar )
grammar_object.setResultsCallback( gotRecognition )
grammar_object.activate( “start”, 0 )
natlink.setMicState( “on” )

natlink.waitForSpeech()
natlink.natDisconnect()
```

MARIO Speech2Text component

- The component includes two modules
  - **Speech2Text Engine:** sends the recognized text through *Marvin Event Bus* (cf. Ability manager component)
    - Intercepts the text recognized by DNS
    - Cleans the text removing unwanted keystroke sequences inserted by DNS
    - Encodes the text in the uft8 format
    - Sends the text to the *speech2text* topic on the Event Bus managed by *Marvin*
  - **Speech2Text Controller:** controls the state of the microphone
    - Subscribes to the *s2tstatecontroller* topic on the *Event Bus* for state change commands from UI or other components
    - Once it gets a on/off command:
      - Turns the microphone on/off through the natlink APIs
      - Notifies the state change sending echoing the same message in the s2tstate topic of the *Event Bus*

Speech2Text component

Status: the wrapper has been coded and it is able to get the text from DNS and interact with the UI component to start and stop speech recognition

The software code of this component is available in the MARIO svn repository under the path trunk/r2m/Speech2Text

Natural Language Understanding (NLU)

NLU

Requirements

- To maximise naturalness of interaction
- To understand user requests and commands
- To answer simple and complex questions
- To interpret answers to CGA questions
- To tag objects with people, places, events, and sentiment from user vocal input
- To support Italian and English

State of the art

- Question Answering over linked data
- Statistical learning
- Linguistic Frames
- Grammar-based NL interaction
- Knowledge representation
- Frame-based statistical learning has better performance than grammar-based approach [7]
- ECHORD EU Project: understanding focused on spatial commands [8]
- IBM Watson [9]

NLU components

Semantic correction

Machine reading and interpretation (Ludwig)

Interacting with People with Dementia (PwD)

People with Dementia (PwD)
We base the NLU development on two main hypotheses as far as specific requirements for natural language (NL) based interaction with PwD are concerned.

Both hypotheses are based on expected consequences of memory loss, which is one of the most frequent and evident cognitive impairment affecting interaction with PwD

- H1: sometimes PwD may forget terms for referring to objects, locations, people, etc., hence they can frequently make requests with partial information
- H2: PwD may repeatedly and frequently forget how to interact with MARIO or what is a recent posed question or an activity they started

We expect to collect evidence of additional specific requirements during the pilot trials and adapt the NLU component to address them as much as possible

Interacting with PwD

According to H1 and H2, main problems/challenges

- **Interacting with PwD** needs more flexible understanding than with people without cognitive impairments
- *Flexible* means that the robot has to be able to handle partial or incorrect information
- Our approach tries either to **guess the missing information** or to **guess the best matching meaning** with partial information
- **Meaning** is related to data available in the robot's knowledge base (including background knowledge from the web of data cf. *Framester*)
- Creation of datasets instantiating relevant frames/relations describing daily life events and common knowledge

Interacting with PwD

Our approach

- **Semantic correction:** misspelling or incomplete data are filled according to MARIO's knowledge base data (which includes personal as well as common knowledge data)
- **Multilayered** interpretation strategies (from shallow to deep NL processing; different matching strategies)
- Formalised **frame semantics** and **web-scale** lexical and world-fact integrated background knowledge (see *Framester* Deliverable 5.1)
- **Building specific datasets** (to integrate with *Framester*) encoding relevant relations (frames) for daily life events, e.g. object/location, by using existing resources and distributional semantics

Evaluation

- **Comparing** with SOA (e.g. Robocup@Home [10] datasets)
- On the field, by means of collecting feedback during **Pilot trials**

Semantic Correction component

This component is meant to minimise errors (e.g. misspelling) of the input text before it is passed to further processing within the natural language understanding component

It performs a pre-processing step that takes into account the data stored in the MARIO knowledge base

It can identify candidate missing entities in partial input

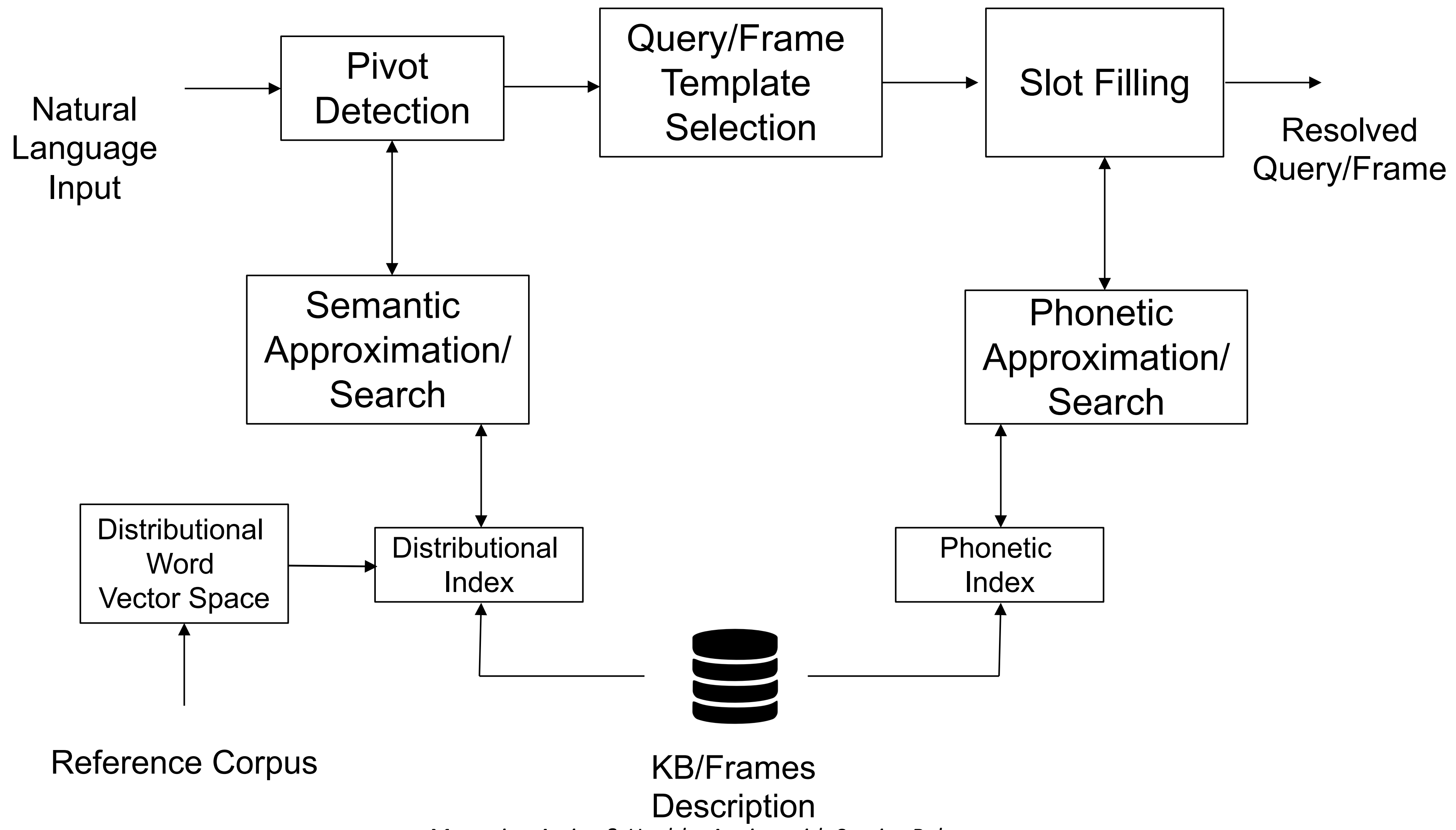
Status: the component has been developed and tested separately, its integration is ongoing work (see details in the following slides)

Requirements

R1: the component addresses misspellings from the Speech to Text (S2T) component using the knowledge from the MARIO background knowledge (see Deliverable D5.1)

R2: It supports multiple languages

Software architecture description



Pivot [M1]

A pivot is a heuristic estimation of the entity in the KB which most likely matches the terms in the natural language input

With the semantic pivoting we have a heuristic method over the quality of possible alignments between input terms and KB entities

This is used to maximise the matching probabilities under a 'mis-listening' scenario

Exploiting pre-defined or recurrent queries

With regard to the query/frames, we are assuming (together with the frames) the case where we could have pre-defined SPARQL queries for the robot (e.g. 'When was the last time that my son visited me?')

In this case we use the heuristics to detect the mappings to the queries

Software architecture description

Semantic pivot detection: Detects the most reliable alignments between the input text and the background knowledge (frame)

Query/frame template selection: Selects candidate queries based on the selected pivots

Semantic approximation: Uses a distributional word vector model (Word2Vec/Skipgram) [5] to support lexical/abstraction-level semantic approximations

Phonetic approximation: Uses the *double metaphone* algorithm [6] to support a phonetic approximation of the input elements

Slot Filling: Resolves the remaining elements of the query/frame using the constrained pivoted subspaces and the semantic and phonetic approximations

Preliminary Results

Development of the first version of the semantic correction component

Evaluation of the platform using simulated misspellings over large open domain knowledge bases and query sets

Evaluation

The platform was initially evaluated using as an experimental setting, the DBpedia dataset with the Question Answering over Linked Data (QALD-4) test collection. The QALD-4 test collection contains 200 natural language queries which are mapped to SPARQL queries over DBpedia

The approach was evaluated by training the model using a ASR query generator (ASR-QG). The simulator takes a query from the test collection, expanding it with a set of WordNet synonyms and generating random word misinterpretations. The misinterpretations are generated after a small set of random editions are produced in a query string which is cast to a similar string from the lexicon (the combined WordNet + DBpedia lexicon) using Levenshtein distance. A set of 4000 queries were generated which were split 80% for training and 20% for test

Evaluation

Different machine learning methods were compared for the semantic pivoting method, including the Multi-Layer Perceptrons (MLP), Long Short-Term Memories (LSTM), Random Forest using different combinations of the following features: lexical features, POS tags, distributional density, IDF (inverse document frequency) of the query term, mapping semantic relatedness

From the different experiment settings, random forest using POS Tags, semantic relatedness and distributional density proved to achieved higher accuracy, achieving 85.18% accuracy on the test set

Further details in D5.4.

Next Steps

Integration and evaluation using the MARIO knowledge bases and frames (deadline: December 2016)

Creation of a test collection using utterances from patients with dementia with different levels of background noise for English and Italian (deadline: February 2017)

Repository

The semantic correction component can be found in the MARIO repository at:

/trunk/passau/semantic_corrector

Usage

Correction (command-line version):

python corrector.py --lang “en” --sentences "sentence" .

returns the corrected sentence.

Training (command-line version):

python training.py --lang “en” --kb “kb.nt” --queries “queries.sparql” --api “apis.frame”

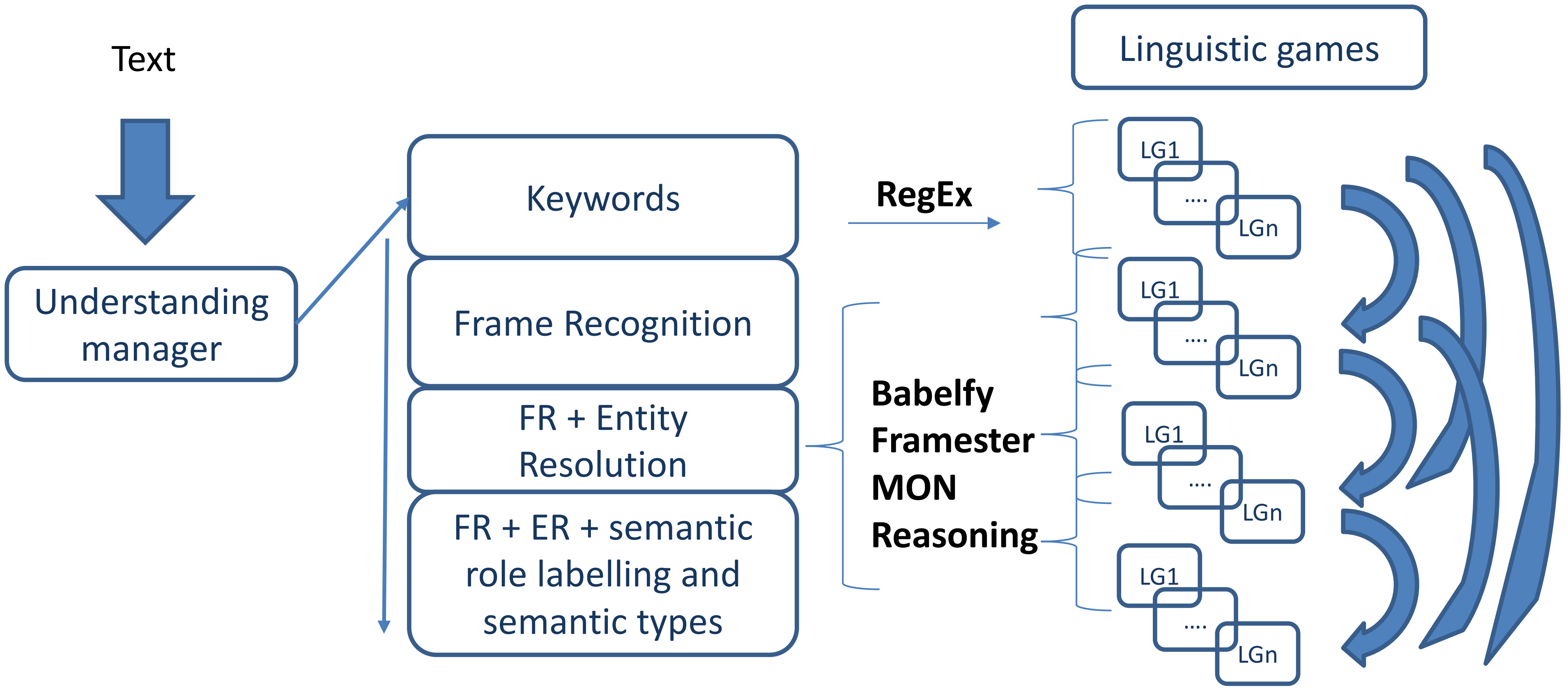
The dependencies are described at the *"requirements.txt"* file, putting the required files into *"data/"* and *"libs/"*.

Ludwig

A python software component based on **linguistic frames** that enables natural language interaction, extensible with additional linguistic abilities

The component implements a layered strategy, as sketched in next slide

Ludwig


Text
Understanding manager
Keywords
Frame Recognition
FR + Entity Resolution
FR + ER + semantic role labelling and semantic types
RegEx
Babelfy
Framester
MON
Reasoning
Linguistic games
LG1
....
LGn

Ludwig

Different strategies are implemented in software modules called *linguistic games*

They span from regular expressions (e.g. interpretation of answers to yes/no questions) to deep frame-based text analysis

They provides NLP services such as word sense disambiguation, frame detection, semantic role labelling, etc.

Ludwig implements matching strategies for associating a request to MARIO’s abilities or to other type of data stored in the knowledge base (e.g. question answering)

Shallow NLP: Regular expression

Paraphrase-based regular expression

- Relying on the http://paraphrase.org/#/ corpus
- Ex. Used for interpreting yes/no answers to CGA questionnaire (cf. Deliverable 4.3)

The software is available in the MARIO svn repository under the path:

trunk/cnr/stlab/ludwig

Paraphrase corpus

The Paraphrase Database (PPDB) [20] is an enormous collection of lexical, phrasal, and syntactic paraphrases. The database is released in six sizes (S to XXXL) ranging from highest precision/lowest recall to lowest average precision/highest recall. PPDB is an automatically extracted database containing millions paraphrases in 16 different languages. The goal of PPBD is to improve language processing by making systems more robust to language variability and unseen words. The entire PPDB resource is freely available under the Creative Commons Attribution 3.0 United States License. PPDB is distributed as plain text files, with one paraphrase rule per line. For English, each line is formatted as follows:

LHS ||| PHRASE ||| PARAPHRASE ||| (FEATURE=VALUE)* ||| ALIGNMENT ||| ENTAILMENT

Where SOURCE is an expression, TARGET is its paraphrase, LHS is the constituent or CCG-style slashed constituent label for the paraphrase pair. ENTAILMENT is an automatically assigned entailment relation (e.g. Equivalence for pairs like couch/sofa, or ForwardEntailment for pairs like dog/animal).

Paraphrase corpus in RDF

In order to make the corpus available for understanding functions and for MARIO's applications we transformed the paraphrase corpus in RDF according to the PPDB ontology that we developed.

The ontology can be found at http://w3id.org/ppdb/ontology

From the paraphrase corpus we extracted only the paraphrases for English and Italian language with an high confidence value.

Matching strategies

Sentence similarity: given two sentences provide a score indicating how close they are

- Reuse of external service based on latent semantic analysis
- The software is available in the MARIO svn repository under the path: *trunk/cnr/stlab/ludwig*

Matching strategies

Ongoing work:

- Implementation of a strategy based on the sentence similarity module for dynamic selection of MARIO abilities based on NL requests (Deadline: February, 2017, to be tested in pilot's 2nd phase, 1st trial)
- Definition of a sentence similarity measure based on distributional representation of word senses and frames
- Creation of two additional corpus for supporting partial information interpretation and question answering:
  - object/typical location relation (by mining DBpedia)
  - Human tasks and procedures (based on refactoring and integration of Hownet [11]) (Deadline January 2017, to be used in pilots' 1st phase, 2nd trial or 1st phase 1st trial)

Deep frame-based NL processing

FRED: existing STLab software for frame-based deep text analysis [12,13]. It has been extended [M2] for

- Supporting adjective semantics
- Integrating Babelfy for performing word sense disambiguation in both Italian and English (we got a free license for research purpose)

Demo available at http://wit.istc.cnr.it/stlab-tools/fred

- Default language is English, for italian use <BING_LANG:it> before the sentence. Always finish a sentence with a dot

Ongoing work:

- Method for mapping FRED graphs to MARIO knowledge base (deadline April 2017, to be tested in pilots’ 2 phase, 2nd trial)

Frame-based ontology matching

This module implements a strategy for aligning ontologies to frames

The aim is to use frames as common interpretation key for text via alignment to the ontologies used for representing knowledge in the MARIO Ontology Network (or any other ontology)

This module is so far to be considered a research proof of concept, hence its integration in the MARIO system will be considered, based on performance results

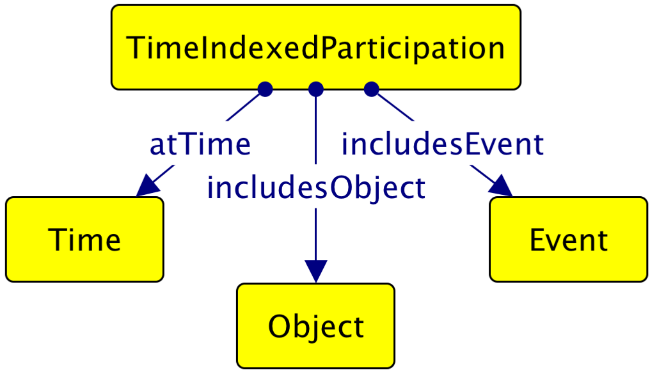
The software is available in the MARIO svn with the path trunk/cnr/stlab/ontomatch. Further details will be provided in next versions of this deliverable

Frame-based ontology matching

Goal Aligning ontologies and frames for bridging information extracted from text delivered with frame semantics and structured information contained in a knowledge base.

Approach Using detour (via Framester) for selecting frames evoked by textual annotations associated with ontology entities and computing the top scoring mappings using structural and semantic text similarity metrics.

1. Ontology entities are interpreted as multi-grade predicates (MGP).



TimeIndexedParticipation (atTime, includesObject, includesEvent)

Frame-based ontology matching

2. Word Sense Disambiguation (through UKB [17] and Babelfy [18]) is performed on textual annotations of ontology entities. To provide the word sense disambiguator with the context for the textual annotation of an ontology entity ***e*** we create a virtual text obtained by concatenating the textual annotations of all the entities $\boldsymbol{e_1}$ in the neighborhood of the ***e*.** For example the context for the class *Time Indexed Participation* is "includes Object includes Time includes event".
We selected the top scoring senses returned by WSD for annotating the labels of the ontology entities.
For example, from the annotations of the class Time Indexed Participation we retrieved the sense participation-noun-1.

Frame-based ontology matching

3. We retrieve in Framester (cf. D5.1) all the frames evoked by the senses selected in the previous step. This is simply done by executing a query on the Framester sparql endpoint (http://etna.istc.cnr.it/framester/sparql).

For example

```
SELECT  ?frame {
  ?frame a fn15schema:Frame .
  ?frame skos:closeMatch wn:synset-participation-noun-1 .
}
```

In this case we retrieved two frames: Participation(Event, Institution, Participant, Time) and Commonality(Commonality, Entities).

Frame-based ontology matching

4. We filter out the frames that show a low textual similarity with the multi-grade predicate extracted from the ontology. To this end for each frame retrieved in the previous step we create a virtual text by concatenating the labels of the frame, frame elements and semantic types. For instance, the virtual text for the frame Commonality is “Commonality entities”. The same is done for the multigrade predicates extracted from the ontology. We compute the Semantic Text Similarity (by using ADW[16]) between the virtual text of the multigrade predicates from the knowledge base and the virtual text of the frames retrieved from Framester. We filter out the frames with a lower score. In our example the textual similarity of Commonality and Time Indexed Participation is lower than the similarity wrt the frame Participation, hence Commonality is discarded.

Frame-based ontology matching

5. We create a mapping between the frames and the MGP extracted from the ontology. For each frame returned by the previous step we compute the mapping with the MGP extracted from the ontology. This step aims at finding the correspondences between the arguments of the MGP (e.g. atTime, includesEvent etc.) and the frame elements of the retrieved frames. The arguments of the MGP are mapped on the most similar frame element. The similarity is provided by computing the semantic text similarity (using ADW[16]) of their labels. For instance, includesEvent is mapped on the frame element Event, atTime is mapped on the frame element Time.

Populating the MARIO KB with prototypical Object-Location relation using distributional semantics

Goal: Equipping MARIO with prototypical knowledge about physical objects and the locations where it is likely to find them. For example, it is usual to find the *dishwasher* in the *kitchen* and in the *utility room.* This knowledge can be use to strengthen the MARIO understanding capabilities with contextual knowledge as well as used for other tasks (e.g. stimulating memory).

Approach: Extracting this relation from a text corpus using distributional semantics.

Relation extraction (RE): A relation is a tuple t = (e_1, .. e_n) where e_i are entities in a predefined relation *r* within a document D. Relation extraction is the task of extracting the tuples t from a document D.

RE with distributional semantics: *Basile et al.* [14] suggested that the relatedness relation encoded in distributional vector representations can be made more precise based on the type of the entities involved in the relation, i.e. if two entities are distributionally related, the natural relation that comes from their respective types is highly likely to occur. For example, the location relation that holds between an object and a room is represented in a distributional space if the entities representing the object and the room are highly associated according to the distributional space's metric.

Populating the MARIO KB with prototypical Object-Location relation using distributional semantics

Procedure The procedure for extracting Object-Location relation is outlined in [14]:

1. **Obtaining a *word vector space* model of the entities of a given corpus**. Word space vectors are abstract representations of the meaning of words, encoded as vectors in a high-dimensional space. A word vector space is constructed by counting cooccurrences of pairs of words in a text corpus, building a large square n-by-n matrix where n is the size of the vocabulary and the cell i,j contains the number of times the word i has been observed in cooccurrence with the word j. The i-th row of the matrix represents the distributional representation of the corresponding word in the corpus. Words that appear in similar contexts often have similar representations in the vector space; this similarity is geometrically measurable with a distance metric such as cosine similarity, defined as the cosine of the angle between two vectors. Alternatively, a precomputed vector space representation can be used. We used NASARI which is a vector space representation for BabelNet synsets (which include WordNet synset and Wikipedia entities).

Populating the MARIO KB with prototypical Object-Location relation using distributional semantics

2. **Selecting vectors representing objects and locations.** In [14] Basile et. al. used a small subset of objects (only those falling under the wikipedia category **Domestic_implements**). By exploiting Framester (cf. D5.1) we were able to select a larger set of wikipedia entities representing physical objects. We proceeded as follows:
    a. We selected from DBPedia (the Linked Open Data version of Wikipedia) the entities that refers to "*generic entities*". Despite being an encyclopedia, Wikipedia contains articles that refers to usual encyclopedic entries (e.g. "Car"), that we called "generic entities", and articles that refers to "specific entities" (e.g. Ford Fiesta). DBPedia does not provide a clear differentiation between specific and generic entities. We devised a method for selecting generic entities from DBPedia. This method relies on the idea that DBPedia's entities that refers to generic entities can be also found in a lexical KB. To this end we used the alignment between DBPedia and lexical KBs aligned by BabelNet (i.e. WordNet, Wikidictionary and OmegaWiki).

Populating the MARIO KB with prototypical Object-Location relation using distributional semantics

b. We filtered the DBPedia generic entities that refers to physical objects. We could not use the DBPedia typing since most of the generic entities are declared belonging to the class owl:Thing (which is the top most generic class in the owl language) or are not typed.
Instead of using DBPedia typing, we used the DBPedia categorization. In fact every DBPedia entity is associated with a wikipedia category. Unfortunately, the taxonomy of wikipedia category is unreliable. To overcome this issue we used an improved version of wikipedia category taxonomy provided by YAGO in which the wikipedia category taxonomy is cleaned using WordNet's hyponomy relations. Hence using this taxonomy together with OntoWordNet we were able to select all the wikipedia categories that falls under the DOLCE class dul:PhysicalObject. The selected categories allowed us to select all the DBPedia's generic entities referring to physical objects.

We are evaluating steps a) and b) using the [15] as baseline.

Populating the MARIO KB with prototypical Object-Location relation using distributional semantics

3. **Computing the similarity between vectors representing objects and locations entities.** Given an object *o* and a location *l* and let v_o and v_l be the vectors associated to o and l respectively, the cosine similarity between v_o and v_l provides a measure of the similarity of o and l. This score is an indicator of how typical is the location l for the object o. Given an object, we can create a ranking of locations with the most likely location candidates at the top of the list. Additionally, an empirical measure commonness of entities could be used to rerank or filter the result to improve its generality. In [14], Basile et al. used a URI counts extracted from the parsing of Wikipedia with the DBpedia Spotlight tool for entity linking. The similarity ranges from -1 (unrelated) to 1 (related).

An example of relation is the following.

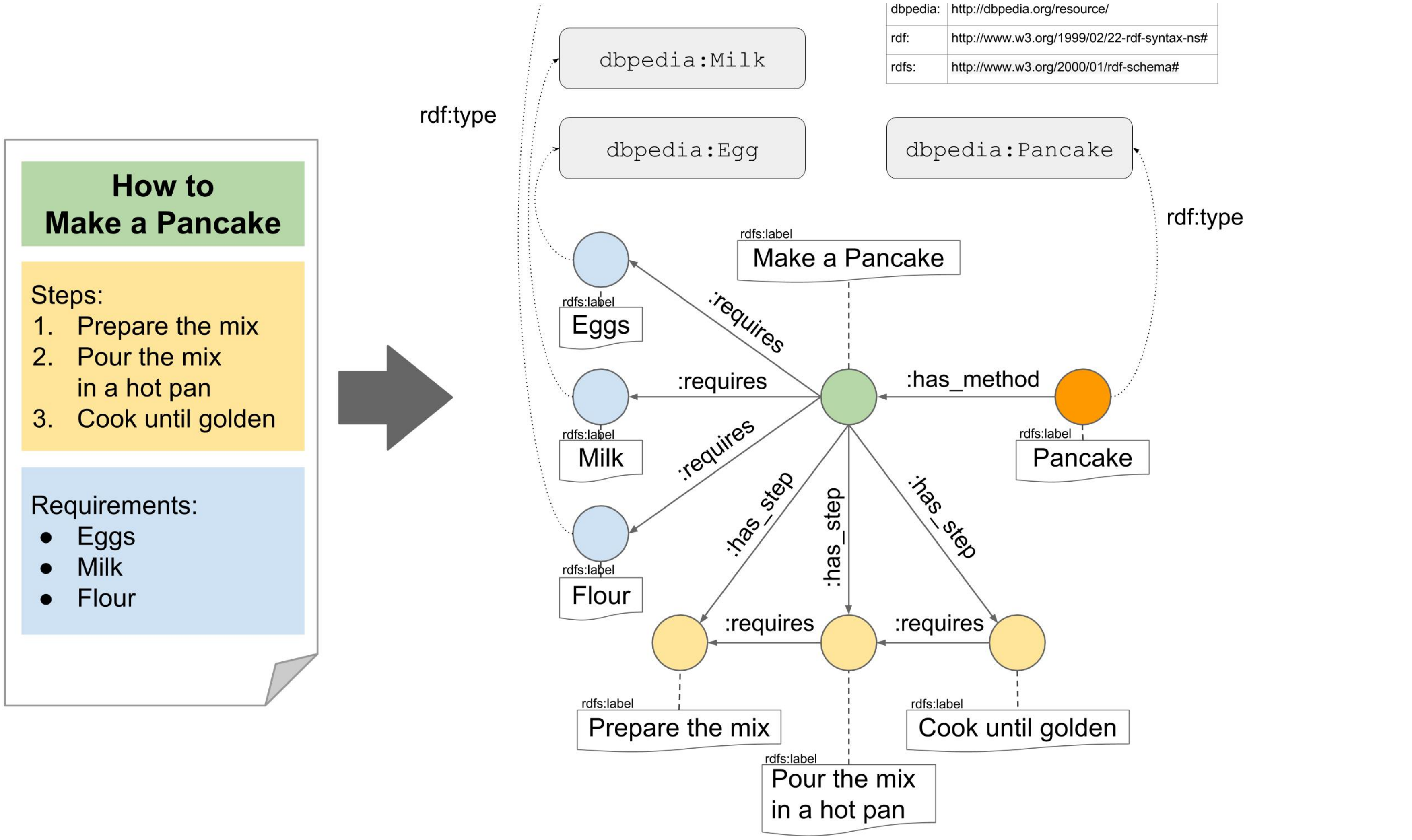
Object	Location	Similarity
Dishwasher	Kitchen	.803
Dishwasher	Laundry room	.788
Dishwasher	Utility room	.763

Populating the MARIO KB with generic procedural knowledge

Goal Equipping MARIO procedural knowledge. For example, to make a pancake someone needs a set of ingredients (Eggs, Milk and flour) and need to perform a series of steps (e.g Mix, Cook etc.). This information could be useful on the one hand to strengthen the understanding capabilities with contextual information (if the use is preparing a pancake is more likely to listen words such as “cook”, “mix”, “milk” etc.). On the other hand, MARIO could use this knowledge to stimulate and guide the user in doing something.

Knowledge source We selected and integrated in the MARIO knowledge base the Human Activities Dataset [19].

The Human activities dataset



How to Make a Pancake

Steps:
1. Prepare the mix
2. Pour the mix in a hot pan
3. Cook until golden

Requirements:
- Eggs
- Milk
- Flour

dbpedia:	http://dbpedia.org/resource/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#

Integration with MARIO KB

Since both the Human Activities dataset and Framester are aligned with the DBPedia Knowledge Base, the two dataset are naturally aligned.
However, only inputs and outputs of procedures contained in the Human Activity dataset are aligned to DBPedia.

We disambiguated the description of the steps with respect to BabelNet and we also extracted the occurrences of the Framester frames to create a machine-readable representation of the steps.

For example, the machine-readable representation of a step of the procedure “How to make homemade sweet bread” is “Combine whisked eggs with milk and mix.”. In this step there is an occurrence of the Framester’s frame Cause_to_amalgamate:

Cause_to_amalgamate(Part_1: dbpedia:Egg, Part_2: dbpedia:Milk)

References

[1] Charles J Fillmore. Frame Semantics and the Nature of Language. Annals of the New York Academy of Sciences, 280(1):20-32, 1976.

[2] Marvin Minsky: A Framework for Representing Knowledge. MIT-AI Laboratory Memo 306, June, 1974.

[3] Aldo Gangemi, Valentina Presutti: Towards a pattern science for the Semantic Web. Semantic Web 1(1-2): 61-68 (2010)

[4] FrameNet: http://framenet.icsi.berkeley.edu/

[5] Mikolov et al., Distributed Representations of Words and Phrases an their Compositionality, NIPS, (2013)

[6] Lawrence Philips, Hanging on the Metaphone, Computer Language, Vol. 7, No. 12 (December), 1990

[7] Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, Roberto Basili, Daniele Nardi: Effective and Robust Natural Language Understanding for Human-Robot Interaction. ECAI 2014: 57-62

[8] European Clearing House for Open Robotics Development http://www.echord.info/

[9] IBM Watson http://www.ibm.com/watson/

[10] Robocup@Home http://www.robocup.org/robocup-home/

References

[11] The Web of Know-How: http://homepages.inf.ed.ac.uk/s1054760/prohow/index.htm

[12] Valentina Presutti, Francesco Draicchio, Aldo Gangemi: Knowledge Extraction Based on Discourse Representation Theory and Linguistic Frames. EKAW 2012: 114-129

[13] http://wit.istc.cnr.it/stlab-tools/fred

[14] Basile, V., Jebbara, S., Cabrio, E. and Cimiano, P.. Populating a Knowledge Base with Object-Location Relations. EKAW 2016. 34-50

[15] Gangemi, A., Nuzzolese, A. G., Presutti, V., Draicchio, F., Musetti, A., & Ciancarini, P. Automatic typing of DBpedia entities. In International Semantic Web Conference (pp. 65-81).

[16] Pilehvar, M. T., Jurgens, D., & Navigli, R. (2013, August). Align, Disambiguate and Walk: A Unified Approach for Measuring Semantic Similarity. In ACL (1) (pp. 1341-1351).

[17] Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. Proceedings of the 12th conference of the European chapter of the Association for Computational Linguistics (EACL-2009). Athens, Greece.

[18] Moro, Andrea, Alessandro Raganato, and Roberto Navigli. "Entity linking meets word sense disambiguation: a unified approach." Transactions of the Association for Computational Linguistics 2 (2014): 231-244.

[19] "The Web of Know-how" http://homepages.inf.ed.ac.uk/s1054760/prohow/index.htm

[20] PPDB http://paraphrase.org

Publications describing research in this task

[M1] André Freitas, Juliano Efson Sales, Siegfried Handschuh, Edward Curry: How hard is the Query? Measuring the Semantic Complexity of Schema-Agnostic Queries, In Proceedings of the 11th International Conference on Computational Semantics (IWCS), London, 2015

[M2] A. Gangemi, V. Presutti, D. Reforgiato Recupero, A. G. Nuzzolese, F. Draicchio, M. Mongiovì: "Semantic Web Machine Reading with FRED". Semantic Web Journal (accepted) http://www.semantic-web-journal.net/content/semantic-web-machine-reading-fred-1