Tools and Methodology for Converting Natural Language into RDF Representations

Olga Loia
MSc in Web Intelligence
International Hellenic University
Thessaloniki 57400, Greece
olgaloia@yahoo.gr

Eleni Kamateri
MSc in Web Intelligence
International Hellenic University
Thessaloniki 57400, Greece
ekamater@hotmail.com

Pavlos D. Vasileiadis
MSc in Web Intelligence
International Hellenic University
Thessaloniki 57400, Greece
ait12019@ait.ihu.gr

ABSTRACT

Recently, there is an increased interest on approaches, methodologies and tools converting natural language into machine readable representations, such as RDF format. Although several steps have been made towards Natural Language Processing (NLP) domain, the machine intelligence for understanding natural language is still a topic of high research interest and its further exploration is going to bring significant benefits to several sub-tasks such as machine translation and questions answering. Natural language processing and natural language understanding are the usual approaches to transform the textual content from being unstructured and ambiguous, into structured and unambiguous. The current article presents the pipeline of linguistic processing and knowledge extraction techniques that are combined together in order to extract structured knowledge from natural language textual content and populate a knowledge base with quality triples. Moreover, it gives an overview of existing knowledge extraction tools found in the literature. Last, a short discussion and comparison of these tools is provided, and the article concludes with a brief summary.

Keywords

Natural language processing, natural language understanding, knowledge extraction, RDF representation, tools

1. INTRODUCTION

In our days, an increasing amount of unstructured text is generated on the Web. It is true that semantic technologies and NLP techniques can serve an important role in mapping and linking of text to formal knowledge representations for achieving efficient retrieval and management. However, unambiguous processing of natural language, in particular, the ability to capture complex knowledge statements is far from finding a solution. The efforts to extract more complex insights from the text persistently require other NLP approaches.

A key approach on this issue is producing quality linked data and ontologies from text, which can enhance the accuracy of the question answering and the machine reading applications. A combination of language and knowledge processing can produce a pipeline that is capable of extracting knowledge in structured forms, to generate quality triples for the Semantic Web through two main tasks [1]: a) linguistic processing tasks, and b) knowledge extraction tasks. The linguistic tasks are mainly useful for the pre-processing of the text, and the knowledge extraction tasks can prove useful for processing the text and generating a form that can be used to extract the triples without losing semantics.

The rest of this article is structured as follows. Section 2 describes the pipeline of language and knowledge processing tasks for extracting structured knowledge from natural language text. Section 3 presents knowledge extraction tools. Last, section 4 concludes the work giving a brief overview of the discussed topics.

2. LANGUAGE AND KNOWLEDGE PROCESSING PIPELINE

The language and knowledge processing pipeline for knowledge extraction consists of a series of linguistic processing and knowledge extraction tasks/components that can be applied the one after the other or simultaneously.

First in the pipeline goes the step of linguistic processing tasks. Early NLP approaches were mainly rule-based techniques, because the computational power was not sufficient enough to use machine learning or statistical methods. Later NLP approaches adopted the use of statistical models in order to increase the performance of some demanding linguistic processing tasks such as Part-of-Speech (POS) tagging.

Thereafter, it comes the knowledge extraction tasks which focus on the extraction of machine-readable knowledge from the textual content. Both rule-based and statistical-based techniques can be seen in the literature

In the following sub-sections, both linguistic processing and knowledge extraction tasks are analyzed.

2.1 Linguistic processing tasks

The linguistic processing components are the first tasks taking place in the language and knowledge processing pipeline. These components are used for cleaning and preparing the text before sending it to the knowledge extraction components.

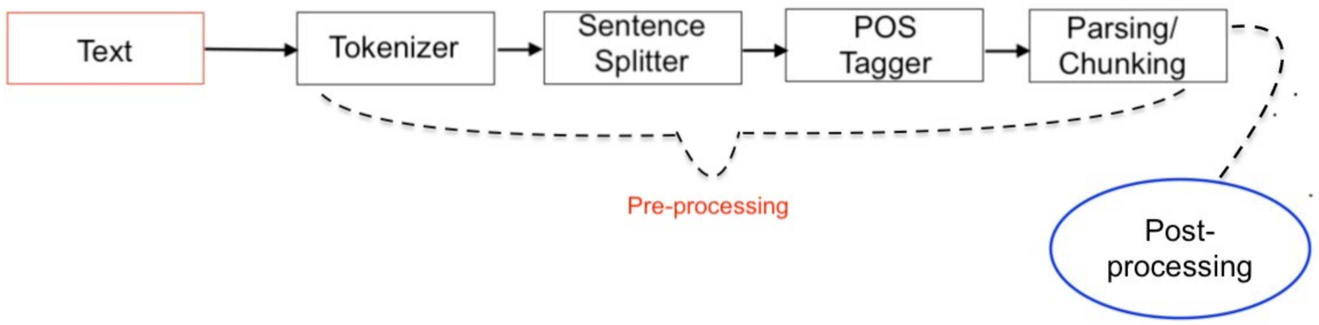


Fig. 1: A typical pre-processing NLP pipeline.

Text cleansing involves splitting the text into tokens (tokenization), removing punctuation, extra spaces, text between brackets, and even unwanted text, including stop words, etc. On the other hand, preparing the text involves techniques such as splitting sentences into segments (segmentation), assigning a category to each token in the sentence (POS tagging), stemming/lemmatization to identify the root of a word and parsing (syntactic analysis) of a sentence.

Below, an overview of the most commonly used linguistic processing tasks is provided.

2.1.1 *Stop Word Handling*

The term "stop word" refers to the most common words in a language such as function words, like "the", "is", "at" and others, and marks like dot, comma and semicolon. While search engines remove function words, linguistic processing tools should identify stop words but not remove them because they can contain invaluable information about the relationship between different tokens.

In general, we can remove stop words while performing the following tasks:

- Text Classification
    1. Spam Filtering
    2. Language Classification
    3. Genre Classification
- Caption Generation
- Auto-Tag Generation

While, we can avoid stop word removal:

- Machine Translation
- Language Modeling
- Text Summarization
- Question-Answering problems

Here's a basic list of stop words:

a about after all also always aman and any are at be been being but by came can cant come could did didn't do does doesn't doing don't else for from get give goes going had happen has have having how i if ill i'm in into is isn't it its i've just keep let like made make many may me mean more most much no not now of only or our really say see some something take tell than that the their them then they thing this to try up us use used uses very want was way we what when where which who why will with without wont you your youre

2.1.2 *Tokenization*

Tokenization is the process of splitting a sentence into smaller units (tokens) using a separator or a whitespace (or ML techniques). In case the input text is a natural language question, the sentence ends either with a question mark or with a period, so simple whitespace splitting tokenizers are often not good enough. Depending on the task to solve, part of the tokenization process can be split on punctuation marks or deleting them.

The tokenization process is crucial as most of the subsequent components in the pipeline cannot process the whole text without being tokenized. Hence, a low accuracy tokenizer can affect the results of the whole pipeline.

Here is an example of the tokenization process performed on a phrase:

Input: Friends, Romans, Countrymen, lend me your ears;

Output:

Friends	Romans	Countrymen	lend	me	your	ears

2.1.3 *Segmentation*

A common NLP task, usually used in standard NLP pipelines, is the sentence segmentation, also called sentence splitting. As seen in Figure 2, segmentation is used to split the given text into separate sentences. The task is based on rules that enable punctuation detection such as full stops, question marks and determine if they indicate the end of a sentence. A list of abbreviations is used by some of these rules such as to distinguish a full stop intended for an abbreviation from full stops to end a sentence. Moreover, some rules used by splitters are implemented to identify and handle structures of different sentences such as bullet lists, titles and addresses which can cause many errors in the segmentation task. For the aim of improving the accuracy of sentence splitters they can be trained on different types of text such as wiki text, hash tags, etc.

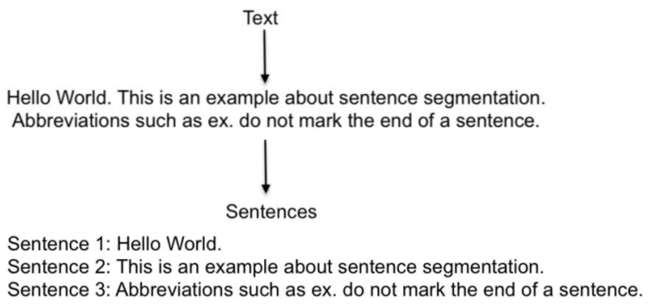

Text
Hello World. This is an example about sentence segmentation.
Abbreviations such as ex. do not mark the end of a sentence.
Sentences
Sentence 1: Hello World.
Sentence 2: This is an example about sentence segmentation.
Sentence 3: Abbreviations such as ex. do not mark the end of a sentence.

Fig. 3: An example output of the segmentation process.

2.1.4 *Part Of Speech (POS) Tagging*

POS Tagging refers to a category of words with similar grammatical properties. All languages have the POS tags noun and verb. The tagging is based both on the token itself and its context. For example, in the sentence "Give me your answer", "answer" is a Noun, but in the sentence "Answer the question", "answer" is a verb.

For POS Tagging, it is important to tokenize the text and identify end-of-sentence punctuation. The information produced by the POS tagger is used by the next linguistic processing tasks in the pipeline, which are the lemmatization and the dependency tree parsing.

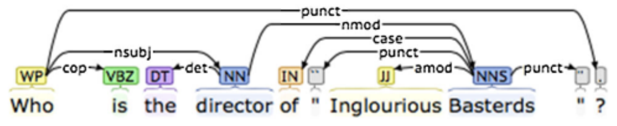


Fig. 4: PoS tags and dependency tree using Stanford CoreNLP [2].

There are different techniques for POS Tagging:

- Lexical-based methods: This method assigns the POS tag the most frequently occurring with a word in the training corpus.
- Rule-based methods: This method assigns the POS tags based on rules. For example, we can have a rule that says words ending with "ed" or "ing" must be assigned to a verb. Rule-Based Techniques can be used along with Lexical Based approaches to allow POS Tagging of words that are not present in the training corpus but are there in the testing data.
- Probabilistic methods: This method assigns the POS tags based on the probability of a particular tag sequence occurring. Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs) are probabilistic approaches to assign a POS Tag.
- Deep learning methods: Recurrent Neural Networks can also be used for POS tagging.

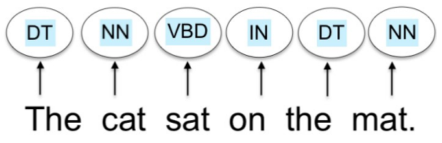


Fig. 5: An example output of the POS tagging process.

2.1.5 *Stemming and Lemmatization*

Both stemming and lemmatization are aiming to reduce inflectional and derivationally related forms of a word to a common base form.

Stemming is attempting to reduce related words of the same stem (i.e. root form of the word) by removing different endings of the words. To succeed this, most stemming algorithms refer to a crude heuristic process that chops off the ends of words. The most widely used algorithm for stemming in English is the Porter's algorithm. It is based on simple rules that are applied to the longest suffix. A weak point of stemming is that the generated stem does not only consist of words with a similar meaning, while same words might have very different meaning and origin. As a matter of fact, in general, stemming increases recall but harms precision.

On the other hand, lemmatization removes inflectional endings and returns the lemma itself, which is either the base form of the word or the dictionary form. For successfully achieving that lemmatization, algorithms usually use a vocabulary and morphological analysis of the words. This way they connect each word with the best deriving term. Lemmatization is usually used together with PoS tagging, which leads to more precise distinctions between the morphological forms of the words. Irregular verbs are translated to their base form. In contrast to stemming, lemmatization can be improved by using context.

2.1.6 *Parsing*

Parsing is the process of analyzing the grammatical structures (syntax) of a sentence so as to acquire the original meaning behind the sequences of words. In most of the cases the parser is based on context-free grammar. Two main directions of how to look at the syntax are available:

The first main direction is constituency syntax and analyzes not only the words but also more complex entities (i.e. constituents). It is also known as statistical parsing. The information produced by the parsing process is normally stored as a syntax tree. It starts with a treebank where all sentences are syntactically annotated (parsed sentences). Then, the frequency of each parse tree is estimated. The generated parse trees are then used to extract a group of corresponding grammar rules with associated probabilities with the aim to define the relative frequency of each rule, which derives the most probable parse in the space against all candidate parses using deduction.

The second main direction is dependency syntax and it looks at the syntactic structures as relations between words. This modern way of parsing attempts to capture the dependency between words instead of identifying the relations between words. The parser works by predicting a sequence of transitions until it approaches the final configuration, these transitions are learned from gold sequences of transitions in a treebank, which are then used to train a multi-class classifier.

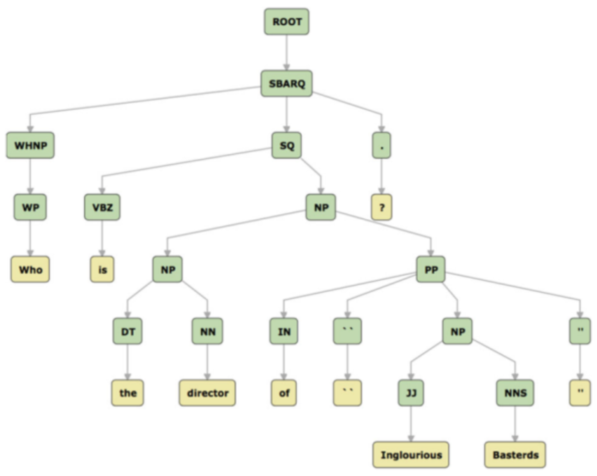


Fig. 6: Constituency tree for the sample question 'Who is the director of "Inglourious Basterds"?' using Stanford CoreNLP [2].

2.1.7 Chunking

Chunking is another NLP process aiming to identify segments of text that are usually syntactically correlated such as noun phrases and verb phrases, and label them as a multi-token sequence. A chunker can replace the parser when a lightweight and shallow analysis is needed over the text. There is a variety of chunkers according to the use required, based on the definition of the relevant chunks of text. For example, as we can see in Figure 6, the noun phrase (NP) can be defined in the chunker as a consecutive sequence string that combines between one or more determiner(s), adjective(s), and noun(s). However, another application may require to be defined so as to include an extra prepositional phrase or relative clause. On the other hand, verb phrase chunkers may also differ from one application to another. For instance, the chunker may recognize modal verbs, infinitive verbs and negative verbs based on the application needs. Learning-based model chunkers can be used in cases that sophisticated text in specific domains is provided. The reliability of these chunkers is further improved by training them on each specific domain text as they will get better results than rule-based chunkers. For instance, in the sentence "I bought the baby food" a rule-based chunker would not be able to distinguish whether the phrase "baby food" is a single NP which represents a compound noun or "the baby" and "food" are two independent NPs.

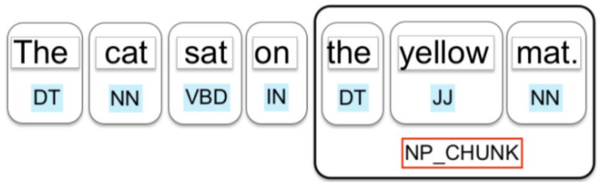


Fig. 7: An example output of a Noun Phrase chunker.

2.2 Knowledge extraction tasks

After the low-level linguistic processing tasks where the main focus is on the syntactic analysis of the text, knowledge extraction tasks offer high-level analysis of the textual content as they focus on deriving semantics from text. These tasks are also fundamental for the automatic development of ontologies.

Below, an overview of the most commonly used knowledge extraction tasks is provided.

2.2.1 Co-reference Resolution

This task is connected with the process of connecting the different ways or the different mentions of expressing the same entity within the text. For example, it can happen due to the use of pronouns for expressing an aforementioned entity.

Different methods have been proposed to address this problem based on the complexity of the corpus. The first method is the rule-based approach, which filters the entities, taking into account their semantic information, and then determines the entities that have the highest probability to be coreferent. There are several co-reference resolution tools that are based on rules, such as ANNIE's Orthomatcher [3], Stanford Coref tool [4] and SANAPHOR [5]. The latest improves the results from Stanford Coref by mapping the output to DBpedia ontology [6]. Other methods include the measurement of the distance between two entities and decision tree checks based on pre-defined attributes.

2.2.2 Named Entity Recognition (NER) and Classification

Named Entity Recognition is the extraction of Named Entities, such as persons, locations, or temporal expressions. The task is divided into the recognition of entities and the mapping to the relevant class, which is called classification. The NER and classification task can be applied using three different methods: rule-based approaches, machine learning approaches and hybrid approaches. Rule-based approaches which are the most widely used rely on gazetteers with regular expressions that cannot generate good results due to the ambiguity of the natural languages. On the other hand, machine learning approaches perform better results but require vast amounts of annotated training data.

2.2.3 Entity Linking and Semantic Annotation

The entity linking, or else the entity disambiguation, is part of the semantic annotation, which deals with the annotation of ambiguous entities in the text with a link to a unique entity in the ontology or Linked Open Data (LOD) resources such as DBpedia [7] [8], Freebase [9], YAGO [10]. Entity linking is the next module in the NLP pipeline that usually comes after the NER and classification. Entity linking is challenging since all the LOD resources include several mentions of the same entity in the knowledge base. So, there is a candidate selection module to determine all the possible entries for a given entity in the text, then a reference disambiguation module, which uses a probabilistic model to select the target entity with the right URI, based on some contextual data and ontology information.

2.2.4 Term Extraction

Unlike the NER task that focuses on identifying generic entities spanning all domains, the text extraction deals with entities that vary across domains and applications. There are different approaches for term extraction, however, they follow the same pipeline except for the way they rank the candidate term in each corpus. The initial module in the pipeline applies linguistic preprocessing tasks (tokenization, segmentation, POS tagging, and NP chunking) to recognize and filter the candidate terms, and then various grammar rules are applied to restrict chunks such as noun phrases and stop words. The last module is used to rank each term in the corpus, and it can use two approaches. The first approaches are the distributional knowledge approaches, which typically relies on frequency-based measures to estimate how important a certain term is to a document in a corpus. The latest are the contextual knowledge approaches, which use contextual knowledge to produce weights to help with the term ranking.

2.2.5 Relation Extraction

The relation extraction part comes after having the relevant terms and entities extracted from the text, to understand how these entities are related. There are different approaches for relation extraction, some of them are developed to focus on relations between lexical items, and others focus on relations between concepts. Similarly, with previous knowledge extraction tasks, there are several relation extraction methods: rule-based approaches, bootstrapping approaches (or else semi-supervised), supervised approaches and unsupervised approaches.

3. KNOWLEDGE EXTRACTION TOOLS

In this section, we present several state-of-the-art knowledge extraction tools performing knowledge extraction and ontology population.

3.1 FRED

FRED [3] [4] is an online tool that implements a method for ontology learning and population in the Semantic Web. The major difference compared with existing approaches is that it does not rely on machine learning methods. Being deployed as a web service means that it minimizes computing time is of primary importance. It implements a modular, highly interoperable and customizable architecture aiming to ensure reusability by other applications and extensibility.

The FRED framework is constituted by four main components:

- The *Boxer* performs deep parsing of natural language text including frame-detection and provides an output in Discourse Representation Structure (DRS).
- The *communication* component realizes a lightweight HTTP server based on Restful architecture, which is in charge of publicly exposing APIs for querying the system. It takes a language text and some optional parameters as input and returns an ontology in OWL/RDF construct that is internally well-connected and linked-data-ready.
- The *refactoring* component transforms Boxer output in a form to be passed to the re-engineering component, which is responsible of implementing the semantic transformation from the domain of the Discourse Representation Theory to OWL;
- The *re-engineering* component implements all the required translation and heuristic rules.

The first component is implemented in Prolog and last three components in Python.

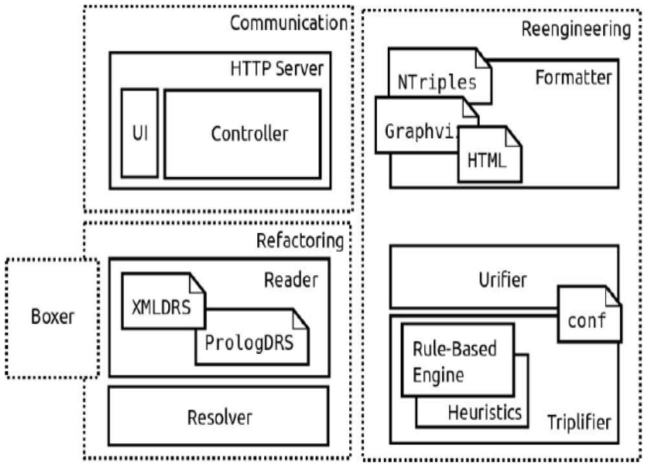


Fig. 8: An overview of the FRED architecture.

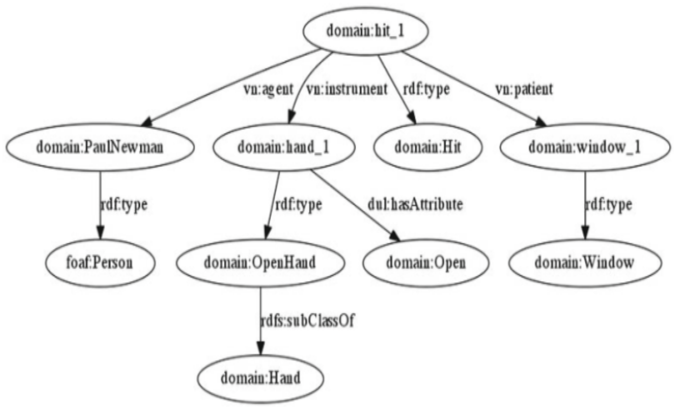


Fig. 9: FRED RDF graph for the sentence "Paul Newman hit the window with an open hand".

3.2 LODifier

LODifier [13] is an approach that combines deep semantic analysis with NER, word sense disambiguation and controlled Semantic Web vocabularies in order to extract Named Entities and relations between them from text and to convert them into an RDF representation which is linked to DBpedia and WordNet [14].

After tokenization, mentions of entities in the input text are recognized using the NER system Wikifier and mapped onto DBpedia URIs. Relations between these entities are detected using the statistical parser C&C [15] and the semantics construction toolkit Boxer [16], which generates discourse representation structures (DRS). Thereafter, the text is lemmatized and words are disambiguated to get WordNet mappings. The RDF graph is then created by further processing the Boxer DRS output, transforming it into triples. Finally, it is enriched with the DBpedia URIs (to link its entities to the LOD cloud) and the WordNet sense URIs (to do the same for the relations).

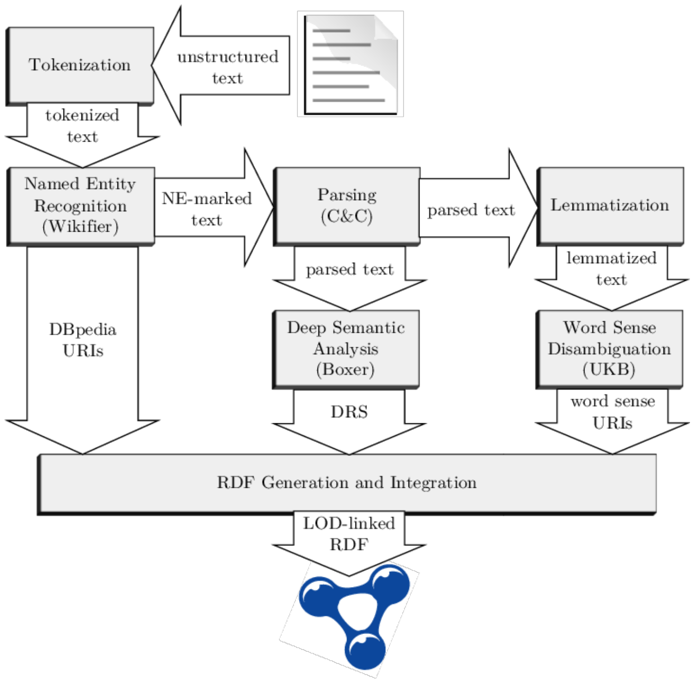


Fig. 10: An overview of the LODifier architecture.

3.3 KNEWS

KNEWS [17] is a pipeline of NLP tools that receives as input natural language text and returns as output knowledge in a machine readable format. More specifically, the tool's output can be frame-based knowledge in the form of RDF triples or XML, including the word-level alignment with the surface form, as well as first-order logical formulae.

The main components of the KNEWS pipeline system are:

- a semantic parser
- a word sense disambiguation module and an entity linking module working together.

KNEWS works by running these components separately on a text, then it aligns the output of the semantic parser to the output of the other two modules (see Figure 9).

3.3.1 Semantic Parsing

The semantic parsing module receives text as input and provides a formal representation of text's meaning as output. To succeed this, KNEWS employs the C&C tools and the Boxer. The C&C tools [15] are a pipeline of statistical NLP tools including a tokenizer, a lemmatizer, Named Entity and POS tagger, and a parser that creates a Combinatory Categorial Grammar (CCG) representation of the natural language syntax. The Boxer [16] is a rule-based system that builds an abstract meaning representation on top of the CCG analysis. Such structures contain, among other information, predicates representing the roles of the entities with respect to the detected events, e.g., event(A), entity(B), agent(A,B) to represent B playing the role of the *agent* of the event A.

3.3.2 Word Sense Disambiguation and Entity Linking

KNEWS uses WordNet to represent concepts and events, DBpedia to represent Named Entities, and FrameNet's frames [18] to represent events, integrating the mapping with the WordNet synsets provided by FrameBase [19]. The thematic roles used by Boxer is taken from VerbNet [20], while KNEWS employs the mapping provided by SemLinks [21] to link the thematic roles to FrameNet roles. By linking the discourse referents representing concepts to WordNet synsets, entities to DBpedia and events to FrameNet frames, KNEWS is able to extract semantic representations from natural language text and link them to Linked Open Data knowledge bases, which is important in order to provide knowledge representation and automatic reasoning.

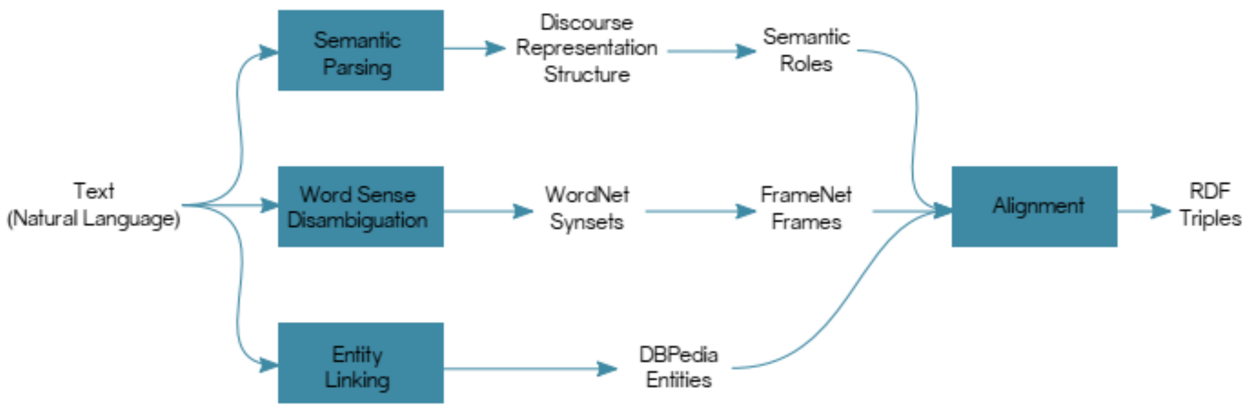


Fig. 11: An overview of the KNEWS architecture

3.3.3 *Outputs*

In the output, KNEWS can provide three output modes:

- **Frame-based Semantics:** Frame instances are sets of RDF triples that contain a unique identifier, the type of the frame, the thematic roles involved in the instance, and the concepts or entities that fill the roles.
- **Word-aligned Semantics.** The word-aligned semantics output mode is similar to the previous one with the difference that it contains as additional information the alignment with the text.
- **First-order Logic.** In the third output mode, KNEWS is able to generate first-order logic formulae representing the natural language text given as input.

```
fb:fi-dc59afa6 a fb:frame-Operate_vehicle-drive.v .
fb:fi-dc59afa6 fb:fe-Driver dbr:Robot .
fb:fi-dc59afa6 fb:fe-Vehicle wn:02961779-n .
```

Fig. 12: An example of frame instance, extracted from the sentence "A robot is driving the car."

3.4 SlugNERDS

The Slugbot's Named Entity Recognition for dialogue Systems (SlugNERDS) tool [22] is a Named Entity Recognition (NER) and Named Entity Linking (NEL) tool which leverages the Google Knowledge Graph API in conjunction with the Schema.org taxonomy to identify known entities. The tool is optimized with respect to noisy open domain conversation and is able to perform both discourse and web-based entity linking.

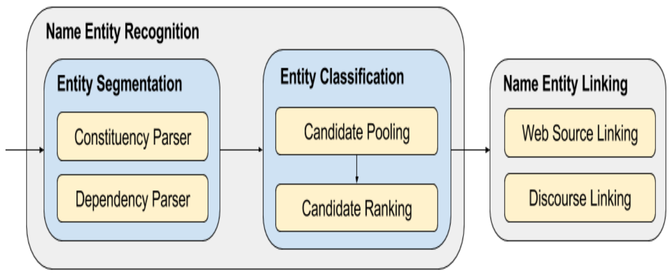


Fig. 13: An overview of the SlugNERDS pipeline.

The SlugNERDS pipeline consists of three modules:

- Entity Segmentation
- Entity Classification

[1] Entity Linking

Entity Segmentation and Entity Classification perform Name Entity Recognition while Entity Linking works on the recognized entity.

3.4.1 *Entity Segmentation*

In order to refine the list of candidate strings to query the authors break the text into reasonable chunks. To do this, they utilize a two-pass approach. First, a constituency tree is constructed using Stanford CoreNLP [2] and a candidate pool is built by collapsing each of the noun phrases, verb phrases, and sentences in the tree. Additionally, sequential noun clusters are collapsed from the dependency parse which have not yet been associated with an entity to create a secondary pool of candidates, so as to include more candidate strings that are ignored by shallow parsing. Last, single pronouns are excluded such as I and me unless they seem extremely contextually relevant.

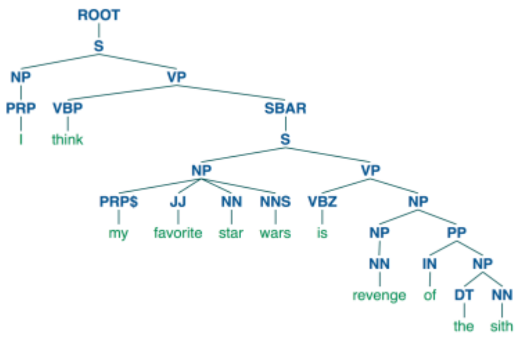
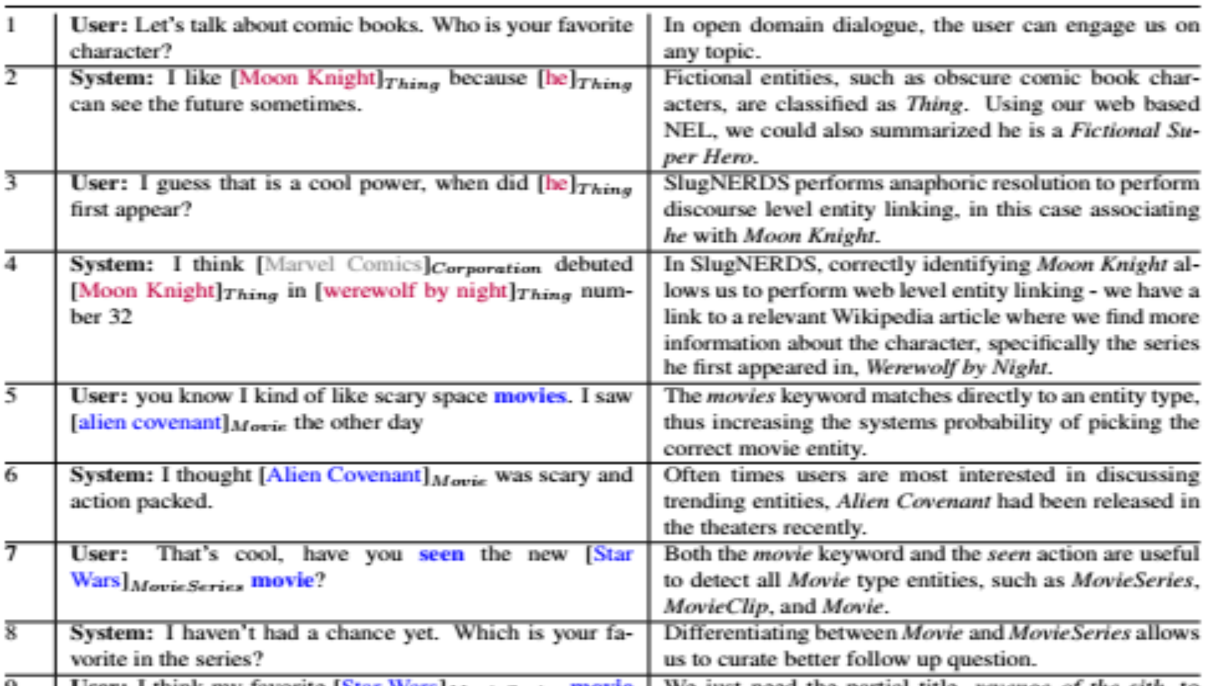


Fig. 14: A sample of a constituency tree.



1	**User:** Let's talk about comic books. Who is your favorite character?	In open domain dialogue, the user can engage us on any topic.
2	**System:** I like [Moon Knight]$_{Thing}$ because [he]$_{Thing}$ can see the future sometimes.	Fictional entities, such as obscure comic book characters, are classified as *Thing*. Using our web based NEL, we could also summarized he is a *Fictional Super Hero*.
3	**User:** I guess that is a cool power, when did [he]$_{Thing}$ first appear?	SlugNERDS performs anaphoric resolution to perform discourse level entity linking, in this case associating *he* with *Moon Knight*.
4	**System:** I think [Marvel Comics]$_{Corporation}$ debuted [Moon Knight]$_{Thing}$ in [werewolf by night]$_{Thing}$ number 32	In SlugNERDS, correctly identifying *Moon Knight* allows us to perform web level entity linking - we have a link to a relevant Wikipedia article where we find more information about the character, specifically the series he first appeared in, *Werewolf by Night*.
5	**User:** you know I kind of like scary space **movies**. I saw [alien covenant]$_{Movie}$ the other day	The *movies* keyword matches directly to an entity type, thus increasing the systems probability of picking the correct movie entity.
6	**System:** I thought [Alien Covenant]$_{Movie}$ was scary and action packed.	Often times users are most interested in discussing trending entities, *Alien Covenant* had been released in the theaters recently.
7	**User:** That's cool, have you **seen** the new [Star Wars]$_{MovieSeries}$ **movie**?	Both the *movie* keyword and the *seen* action are useful to detect all *Movie* type entities, such as *MovieSeries*, *MovieClip*, and *Movie*.
8	**System:** I haven't had a chance yet. Which is your favorite in the series?	Differentiating between *Movie* and *MovieSeries* allows us to curate better follow up question.
9	**User:** I think my favorite [Star Wars] [illegible] **movie**	We just need the partial title, revenge of the sith, to

Fig. 15: An annotated conversation is presented based on Slugbot tool

3.4.2 *Entity Classification*

Once candidate phrases are collected from the entity segmentation phase, each of these phrases are sent as queries in the Google Knowledge Graph API and the top N relevant entities in Candidate Pooling are gathered.

Sometimes it is possible for an entity candidate returned by the query to have the same exact title with different entity types. For example, there are 5 entity candidates with the title "Star Wars: Episode III - Revenge Of The Sith" each with a different entity type (Movie, Video Game, Book, MusicAlbum, and BookSeries).

Since the entities returned by the Google Knowledge Graph may not be an exact match to the query, it gives more flexibility, while introducing some noise. Furthermore, if a user is expected to talk about certain entity types according to the context, increased value is placed on certain entities while penalizing others. Based on these observations, the authors perform a scoring algorithm that maximizes the performance in the Candidate Ranking phase.

Once the entities are all scored, the list is re-ranked and only the top ranked entity for each node are considered, while also pruning away nodes whose top scoring entity was less than a certain threshold (empirically driven). Moreover, overlapping nodes that have candidates are merged. In the last stage, the query/candidate is synced to the internal discourse state representation.

3.4.3 *Entity Linking*

Named Entity Linking consists of two phases, Web Source Linking and Discourse Linking. Web Source Linking performs the linking of a known entity to existing resources on the web while Discourse Linking is focused on the linking of each mention of the entity within the input to the same discourse entity in the internal representation.

3.5 Text2Onto

Text2Onto [23] is a framework for ontology learning from natural language text, which combines machine learning with basic linguistic processing (such as tokenization or lemmatizing and shallow parsing) approaches. Moreover, it is based on the GATE framework for the creation of the linguistic algorithms and models, which provides increased flexibility to the user. Linguistic preprocessing begins with the tokenization and the sentence splitting. The outcome tokens of this process serve as an input for a POS tagger which assigns appropriate syntactic categories to all tokens. Finally, lemmatizing or stemming is done by a morphological analyzer and a stemmer respectively to extract patterns.

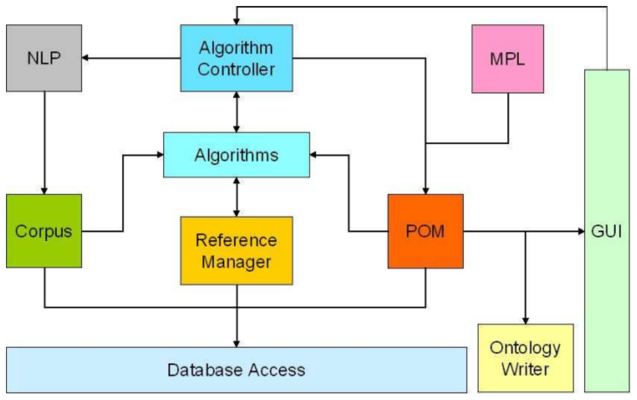


Fig. 16: An overview of the KNEWS architecture

After the basic linguistic preprocessing is done, a JAPE transducer runs over the annotated corpus in order to match a set of particular patterns with the ontology learning algorithms. Whereas the left-hand side of each JAPE pattern defines a regular expression over existing annotations, the right-hand side describes the new annotations to be created. For Text2Onto we developed JAPE patterns for both shallow parsing and the identification of modeling primitives, i.e. concepts, instances and different types of relations. Text2Onto is considered to be the successor of TextToOnto [24], which lacks the flexibility of combining different algorithms and does not facilitate any interactions with the user, which both features have been included in the Text2Onto.

Since both types of patterns are language specific, different sets of patterns for shallow parsing and ontology extraction have to be defined for each language. Because of this and due to the fact that particular processing components for GATE have to be available for each language, Text2Onto currently supports ontology learning only from English texts while it is currently expanded for Spanish and German language.

3.6 SPRAT

The SPRAT (Semantic Pattern Recognition and Annotation Tool) [25] is an ontology generation and population system, that enables the user to create an ontology from scratch or modify an existing one. The system generates new knowledge by linking text to ontologies based on lexico-syntactic patterns.

Its architecture is composed of a number of linguistic pre-processing components (based on GATE) followed by a set of gazetteer lists and the JAPE transducers. More specifically, a tokenizer divides the text into tokens, the sentence splitter divides the text into sentences, the POS-tagger adds POS information to tokens, a morphological analyzer adds morphological information (root, lemma etc.) to tokens and a NP chunker divides the text into noun phrase chunks.

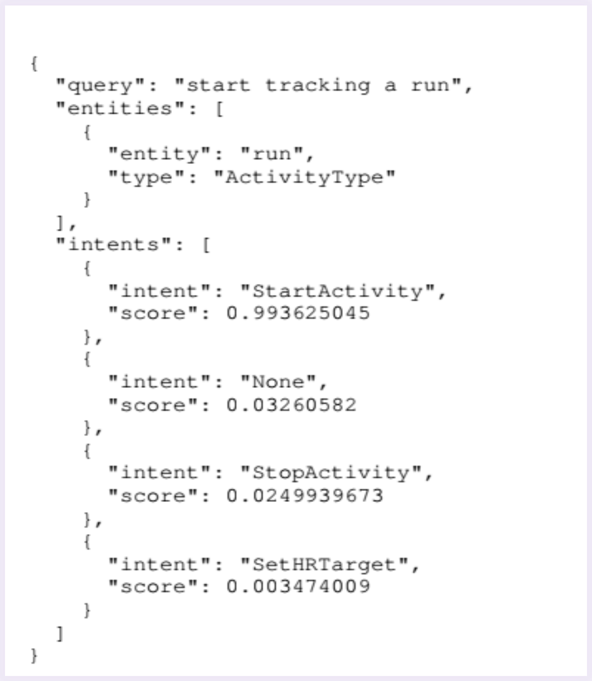
Thereafter, gazetteers look up various items in lists, OntoRootGazetteer (optional) looks up items from the ontology and matches them with the text, based on root forms and JAPE transducers annotates text and adds new items to the ontology.

It differs from Text2Onto as it does not rely on statistical clustering for relation extraction, and it uses more lexicon-syntactic patterns for extracting entities and relations from text. The system accuracy can be improved by refining the patterns and linking it with other NLP resources, such as WordNet conceptual-semantic categories and lexical relations.

3.7 LUIS

The advance of conversational chatbots increased the need for natural language understanding. Due to this, several interfaces has been created lately to accommodate this need. Microsoft LUIS.ai or LUIS.ai or Language Understanding Intelligent Services [26] is an example of these natural language understanding interfaces.

LUIS is a web-based tool for natural language analysis that can be trained for a particular domain. It utilizes machine learning based methods to process and analyze textual utterances. To be able to apply machine learning methods, LUIS breaks down the utterances into smaller pieces called tokens, by a process called tokenization. Then, it represents the extracted semantic information of an utterance as intents and entities. The intent represents the purpose of an utterance while the entities represent the specific details and information. An intent represents what the user wants to achieve with the utterance, the overall goal. The intent is heavily influenced by the context and the domain. An intent of the utterance “I want to go to Germany!” could be that the user wants to book a flight.



```
{
  "query": "start tracking a run",
  "entities": [
    {
      "entity": "run",
      "type": "ActivityType"
    }
  ],
  "intents": [
    {
      "intent": "StartActivity",
      "score": 0.993625045
    },
    {
      "intent": "None",
      "score": 0.03260582
    },
    {
      "intent": "StopActivity",
      "score": 0.0249939673
    },
    {
      "intent": "SetHRTarget",
      "score": 0.003474009
    }
  ]
}
```

Fig. 17: Example JSON response for the utterance “start tracking a run”

However, the same utterance could also mean that the user needs a car or buy a train ticket. Furthermore, every important fact of a sentence is mapped to an entity. Examples of entities are dates, locations or specific products. In the previously mentioned utterance, “I want to go to Germany!”, the entity is Germany. LUIS lets the user create its own sets of intents and entities and presents a decimal value between 0 and 1 indicating how sure it is that the identified intent is the actual intent of an utterance.

4. CONCLUSION

This article was attempted to approach the issue of capturing complex knowledge statements from natural language, which is a challenging problem applying in the domain of the Natural Language Processing (NLP). More specifically, the production of quality linked data and ontologies from natural language can enhance the accuracy of the question answering and machine reading applications. Combining knowledge and language processing produces a pipeline that is capable of extracting knowledge in structured forms, to generate quality triples for the Semantic Web. This requires both linguistic processing and knowledge extraction tasks. The linguistic tasks are mainly useful in the stage of text pre-processing and the knowledge extraction tasks are proved useful for text processing aiming to generate a form that can be used to extract the triples without losing semantics.

More specifically, the article provides an overview of the language and knowledge processing pipeline for knowledge extraction. The pipeline consists of a series of linguistic processing and knowledge extraction tasks/components that can be applied sequentially or simultaneously. The linguistic processing components are used for cleaning and preparing the text before sending it to the knowledge extraction components. These include stop word handling, tokenization, segmentation, Part Of Speech (POS) tagging, stemming and lemmatization, parsing and chunking. This low-level linguistic processing is followed by the knowledge extraction tasks. These include co-reference resolution, named entity recognition and classification, entity linking and semantic annotation, and finally term and relation extraction.

It is following the concise presentation of several state-of-the-art tools that perform knowledge extraction and ontology population. These tools include FRED, LODifier, KNEWS, SlugNERDS, Text2Onto, SPRAT, and LUIS.

5. REFERENCES

[1] F. Corcoglioniti, M. Rospocher, and A. P. Aprosio, “A 2-phase Frame-based Knowledge Extraction Framework,”. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM. 2016, pp. 354–361.

[2] C. D. Manning, J. Bauer, J. Finkel, and S. J. Bethard, “The Stanford CoreNLP Natural Language Processing Toolkit,” In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. 2014, pp. 55–60.

[3] K. Bontcheva, M. Dimitrov, D. Maynard, V. Tablan, and H. Cunningham, “Shallow Methods for Named Entity Coreference Resolution,” In: Chaınes de references et resolveurs d’anaphores, workshop TALN. 2002.

[4] K. Raghunathan et al., “Raghunathan - A Multi-Pass Sieve for Coreference Resolution.” In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics. 2010, pp. 492–501

[5] R. Prokofyev, A. Tonon, M. Luggen, L. Vouilloz, D. E. Difallah, and P. Cudré-Mauroux, “SANAPHOR: Ontology-based coreference resolution,” In: International Semantic Web Conference. Springer. 2015, pp. 458–473.

[6] C. Bizer et al., “DBpedia - A crystallization point for the Web of Data,” In: Web Semantics: science, services and agents on the world wide web 7.3 (2009), pp. 154–165.

[7] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, “DBpedia spotlight: Shedding light on the web of documents,” In: Proceedings of the 7th international conference on semantic systems. ACM. 2011, pp. 1–8

[8] J. Hoffart et al., “Robust disambiguation of named entities in text,” In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics. 2011, pp. 782–792.

[9] Z. Zheng, X. Si, F. Li, E. Y. Chang, and X. Zhu, “Entity disambiguation with Freebase,” In: Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent TechnologyVolume 01. IEEE Computer Society. 2012, pp. 82–89

[10] W. Shen, J. Wang, P. Luo, and M. Wang, “LINDEN: Linking named entities with knowledge base via semantic knowledge,” In: Proceedings of the 21st international conference on World Wide Web. ACM. 2012, pp. 449–458

[11] V. Presutti, F. Draicchio, and A. Gangemi, “Knowledge extraction based on discourse representation theory and linguistic frames,” In: International conference on knowledge engineering and knowledge management. Springer. 2012, pp. 114–129

[12] F. Draicchio, A. Gangemi, V. Presutti, and A. G. Nuzzolese, “FRED: From natural language text to RDF and OWL in one click,” In: Extended Semantic Web Conference. Springer. 2013, pp. 263–267.

[13] I. Augenstein, S. Padó, and S. Rudolph, “LODifier: Generating linked data from unstructured text,” In: Extended Semantic Web Conference. Springer. 2012, pp. 210–224

[14] G. A. Miller, “WordNet : A Lexical Database for English,” Communications of the ACM, vol. 38, no. 11, pp. 39–41, 1995.

[15] J. R. Curran, S. Clark, and J. Bos, “Linguistically motivated large-scale NLP with C&C and boxer,” In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. Association for Computational Linguistics. 2007, pp. 33–36

[16] J. Bos, “Wide-Coverage Semantic Analysis with Boxer,” In: Proceedings of the 2008 Conference on Semantics in Text Processing. Association for Computational Linguistics. 2008, pp. 277–286.

[17] V. Basile, E. Cabrio, and C. Schon, “KNEWS: Using Logical and Lexical Semantics to Extract Knowledge from Natural Language,” In: Proceedings of the European conference on artificial intelligence (ECAI) 2016 conference. 2016.

[18] C. F. Baker, C. J. Fillmore, and J. B. Lowe, “The Berkeley FrameNet Project,” In: Proceedings of the 17th international conference on Computational linguistics-Volume 1. Association for Computational Linguistics. 1998, pp. 86–90

[19] J. Rouces, G. De Melo, and K. Hose, “FrameBase : Representing N-ary Relations using Semantic Frames.”I n Proceedings of ESWC 2015, pages 505--521, 2015.

[20] K. K. Schuler, “VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon,” Diss. Abstr. Int. B Sci. Eng., vol. 66, no. 6, 2005.

[21] M. Palmer, C. Bonial, and D. McCarthy, “SemLink+: FrameNet, VerbNet and Event Ontologies,” In Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (1929-2014) (pp. 13-17).

[22] M. Bowden, K., Wu, J., Oraby, S., Misra, A., & Walker, “SlugNERDS: A Named Entity Recognition Tool for Open Domain dialogue Systems,” in Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[23] P. Cimiano and J. Völker, “Text2Onto A framework for ontology learning and data-driven change discovery,” In: International conference on application of natural language to information systems. Springer. 2005, pp. 227–238.

[24] A. Maedche and S. Staab, “Ontology Learning,” In: Handbook on ontologies. Springer, 2004, pp. 173–190

[25] D. Maynard, A. Funk, and W. Peters, “SPRAT : a tool for automatic semantic pattern-based ontology population.” In: International conference for digital libraries and the semantic web, Trento, Italy. 2009

[26] J. D. Williams, E. Kamal, M. Ashour, H. Amr, J. Miller, and G. Zweig, “Fast and easy Language Understanding for dialog systems with Microsoft Language Understanding Intelligent Service (LUIS),” SIGDIAL 2015 - 16th Annu. Meet. Spec. Interes. Gr. Discourse Dialogue, Proc. Conf., pp. 159–161, 2015.