Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona
Master of Innovation and Research in Informatics(MIRI)

Extracting Temporal Expressions from Unstructured Open Resources

Zagros Valizadeh Ardalan

Advisor: Carme Martín

April 2015

Abstract

Temporal information processing has received substantial attention in the last few years, due to the appearance of temporal expressions in search keywords and importance of them in the internet content.

AETAS is an online tool for converting text into internally well-connected RDF linked data with resolution of temporal expressions. AETAS follows fully SOA architecture and accessible from web-service. It implements a novel approach for semantic representation and linked temporal graphs from natural language sentences.

In this research, we present a demonstration of such a tool combining the normalized temporal expression from context with linguistic frame semantic and create linked RDF graph which time defined as an individual dimension. The tool is based on SUTIME which identify the temporal expressions and normalized them by TIMEX3 standard and WP2, linguistic processor which tokenize, lemmatize and represent the semantic of sentences. The output of AETAS is enriched stored RDF triples that have ability to visualize the events extracted from the web on timeline.

A SPARQL layer has been provided in order to get the results of stored data and visualize them by TIMELINE JavaScript library. The output of the system can be used in different manners; Make a self-recursive system to increase the efficiency of temporal expression tagger and add temporal aspect and dimension into linked data world.

Keywords: Natural Language Processing, Information Extraction, Temporal Information Processing, Temporal Linked Data

Acknowledgment

I feel indebted to my family and my friends for all the help and support during this semester of research leading to this dissertation.

I would also like to thank my advisor and Lluís Padró for their direct supports either in academic, either in motivation aspects. Obviously, my work has benefited directly from the tools that have been developed in the TALP group, and which I made use of for this dissertation, as well as from their cooperation, but more important than that was what I learned from their insights and comments, and also the motivation I gained from their support.

Content Table

Table of Figures

I. Introduction

What are temporal expressions? What is temporal information retrieval? Before starting describing these expressions, let's talks about French Revolution that has been taken from history website: [1]

As the 18th century drew to a close, France's costly involvement in the American Revolution and extravagant spending by King Louis XVI (1754-1793) and his predecessor had left the country on the brink of bankruptcy.

In the fall of 1786, Louis XVI's controller general, Charles Alexandre de Calonne (1734-1802), proposed a financial reform package that included a universal land tax from which the privileged classes would no longer be exempt. To garner support for these measures and forestall a growing aristocratic revolt, the king summoned the Estates-General ("les états généraux")–an assembly representing France's clergy, nobility and middle class–for the first time since 1614. The meeting was scheduled for May 5, 1789; in the meantime, delegates of the three estates from each locality would compile lists of grievances ("cahiers de doléances") to present to the king.

On June 17, with talks over procedure stalled, the Third Estate met alone and formally adopted the title of National Assembly; three days later, they met in a nearby indoor tennis court and took the so-called Tennis Court Oath ("serment du jeu de paume"), vowing not to disperse until constitutional reform had been achieved.

Known as the Great Fear ("la Grande peur"), the agrarian insurrection hastened the growing exodus of nobles from the country and inspired the National Constituent Assembly to abolish feudalism on August 4, 1789, signing what the historian Georges Lefebvre later called the "death certificate of the old order."

Adopted on September 3, 1791, France's first written constitution echoed the more moderate voices in the Assembly, establishing a constitutional monarchy in which the king enjoyed royal veto power and the ability to appoint ministers.

In April 1792, the newly elected Legislative Assembly declared war on Austria and Prussia, where it believed that French émigrés were building counterrevolutionary alliances ….

Events happened in one point of time or happened in an interval time period. As its obvious highlighted texts are temporal expressions in the texts and events belong to them. With a small processing, they can be listed in the table or visualized in the timeline by ascending or descending sorting. Then the user immediately can realize what happened in France revolution.

Temporal Expression	Value	Event
18th century	17XX	American Revolution
fall of 1786	1786-FA	Louis XVI's controller general
since 1614	1614 - NOW	assembly representing France's clergy
May 5, 1789	1789-05-05	meeting was scheduled
June 17	1789-06-17	procedure stalled
three days later	1789-06-20	met in a nearby indoor tennis

[1] http://www.history.com/topics/french-revolution retrieved on 7th April 2015

August 4, 1789	1789-08-04	abolish feudalism
September 3, 1791	1791-09-03	France's first written constitution
April 1792	1792-04	declared war on Austria

The topic of this thesis is how this action can be performed automatically with high quality and how semanticizes all the content with more focus on temporal expressions. Essentially, the goal is to extract structured information about time from unstructured text. This sort of task can be useful in several applications, as presented later in this chapter, but as this example shows, even on its own it can be used to organize data in a way that can make it faster and easier to grasp and visualize.

1.1 Background

Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.

NLP try to behave and make communication with natural text in the way how humans do by creating computational systems and using artificial intelligence algorithms. Natural language processing (NLP) is a theory more than applicant or algorithm; is the theory of finding a model and relationships between natural language elements exactly like how humans handle the languages.

By increasing the web and need of more information that can be processed automatically by machines, natural language processing has become important factor from the beginning of digital era. Accordingly the NLP applications appeared in everyday life and become important topic for computer science research in different areas [Fransico Carapeto, 2012]:

- Perform spell checking and grammar checking,
- Produce a short summary of a document or of a collection of documents (automatic summarization)
- Translate text between different natural languages (machine translation)
- Search the World Wide Web (or any other knowledge base) for specific answers to questions input by a human user (question-answering)
- Extract key-words from documents
- Identify a document's topic, textual genre, author or language, etc. (text classification).

Other kinds of systems and applications in this field are maturing and present great potential.

Most of web content is unstructured data; from news contents, blogs, journals, etc. all of them are written by humans with natural language grammars. But what is used most by machines and current business applications are structured data. Those data saved pretty well in tables with their relationships with another structured data. But what will happen with all of those unstructured data that can't be used in current business application. The huge amount of data that reading them manually and extract information from them is

totally impossible. NLP come into the practice to convert these contextual and unstructured data into well organized and machine-sensible structured information. So the need of NLP becomes deniable and its quality and usages become more and more interesting topic in the World Wide Web.

The world is dynamic, and change is a part of it. Similarly, natural language has several different means to describe when and for how long something happens or stays unchanged. Indeed, a large proportion of the information carried by natural language is temporally circumscribed, as reference to time is widespread in natural language.

Time used in more or less all events in the history and in the normal life. Understanding of them and the development of automatic systems able to extract the temporal meaning of texts have been difficult tasks. Still, the last few years have seen a large amount of research focused on the problem of temporal information processing, with the goal of making NLP systems more sophisticated.

1.2 Temporal Information Processing

Temporal Information Retrieval (T-IR) is an emerging area of research related to the field of information retrieval (IR) and a considerable number of sub-areas, positioning itself, as an important dimension in the context of the user information needs.

The task of temporal information processing can be illustrated with the following paragraph, taken from a news article:

Obama signed the contract **today to let the Internet providers companies** increase **their internet speed from May 2015. Meanwhile congress was focusing on middle east problems.**

A temporal information system while receiving this document, in order to process it, needs to realize when this document has been written. Also need to process the verbs in the sentences to produce the event happen before or after the temporal point. So this elements includes:

1. **Data Creation Time (DTC):** processing a document without reference time make the process very difficult. Normally the reference time is the time when document has been written. For example consider the DTC time of this document on 17-04-2015.
2. **The dates and times referred to in the text:** The word today refers to the document creation date.
3. **The situations that are described in the document:** Signing a contract from Obama is the event that happens in that day, and the companies should do the work from the exact time afterwards.
4. **The temporal relations holding between these situations and dates or times**: The ***Meanwhile*** expression connects two events that happened in the same time. Congress does another event while president is busy for something else.

So if we are going to represent it graphically:

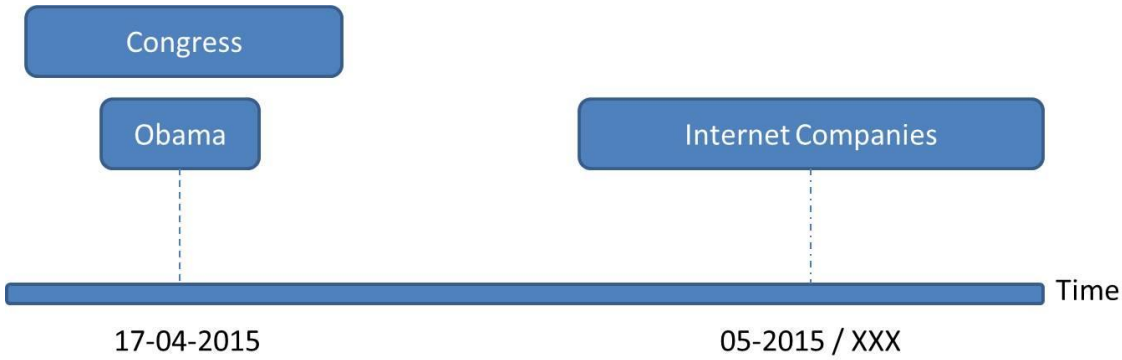

Figure 1: Simple Timeline Example

In this graph, temporal overlap is happened between events. XXX shows the ending time is not clear and as is shown. Internet companies' block continues over the timeline. Also since it's not clear when congress focused on Middle East issues, the start date and end date has not been shown, but it's clear that it passed today!.

1.3 Types of Temporal Information

Temporal expressions mentioned in text documents can be grouped into four types according to TimeML [J. Pustejovsky and J. M. Casta, 2003]:

- Date: ***Today*** I went to cinema
- Time: Let's see each other at ***15:30 afternoon***
- Duration: I'm working on this project ***from last week***
- Set: ***Each Tuesday*** we meet each other in coffee shop.

While duration expressions are used to provide information about the length of an interval, set expressions inform about the periodical aspect of an event. In contrast, time and date expressions both refer to a specific point in time. An interesting key feature of temporal information is that it can be normalized to some standard format. Assuming a Gregorian calendar as representation of time, expressions of time and date can be directly placed on a timeline. A date is then typically represented as YYYY-MM-DD, e.g., the expression "January 25, 2015" is normalized to "2015-01-25". However, the normalization is not always as simple as in this example, and sometimes need many process and calculation to identify when the writer pointed in the content.

Temporal expressions and temporal information has many types to be mentioned in the content. It can be:

- ***Explicit:*** January 2015
- ***Relative***: Next Monday
- ***Implicit:*** new year's eve
- ***Ambiguous***: Fall (Autumn)

According to work of Schilder and Habel [F. Schilder and C. Habel, 2001] explicit temporal expressions refer to a specific point in time but with different dimension. For example 12th April 2015 refers to day level of time, while September 2014 showing a point on month level and can cover the whole month. The key characteristic of explicit temporal expressions is that they can be normalized with without any further knowledge. That is, the expression itself contains all the information needed for normalization and is thus fully specified.

The second type of temporal expressions is Relative. They can't be normalized that much easy like explicit one and it needs the calculation and process the content beforehand. For example, the expression "today" cannot be normalized without knowing the corresponding reference time. This reference time can either be the document creation time or another temporal expression in the document. Typically, in news articles, the document creation time is important and often used as reference time. If we consider the reference date as 17-04-2015, the expression "Yesterday" in The Mobile World Congress has been started from ***yesterday*** in Barcelona, refers to 16-04-2015. Temporal expression can be relative with another temporal expression. For example "Last day of 2015" refers to 31-12-2015 which has calculated by last day with the year 2015. In other types of documents, the reference time is usually represented by another temporal expression in the document. In general, there are many occurrences of relative temporal expressions. While sometimes the reference time is sufficient for normalization, further information is needed if the relation to the reference time has to be identified as well. For example, "on Monday" can either refer to the previous or to the next Monday with respect to the reference time. An indicator for determining the relationship can be the tense of the sentence with future tense and present tense indicating an after-relationship to the reference time and past tense a before-relationship.

Implicit types of temporal expressions are the ones which match with calendar and timeline. These expressions can be anchored on a timeline if a mapping of the expression to its normalized value is available. For example, "New Year's Day 2015" can be normalized to "2015-01-01" since "New Year's Day" always refers to January 1. In addition, there are expressions for which a temporal function has to be applied. "Labor Day", for example, refers to the first Monday in September so that "Labor Day 2009" can be normalized to "2009-09-07" if we know this day to be the first Monday in September 2009.

The fourth type of temporal expressions are those which are not clear is it temporal expression or it's another tense in the sentence. For instance, the word "Fall" as mean as "Autumn" can be temporal expression when it's noun and placing in specific part of sentence or it can be verb which is not temporal expression at all.

Although there are many different ways to refer to a specific point in time, all expressions referring to the same point in time shall be normalized to the same value in the standard format. This normalization process is one of the tasks of so-called temporal tagging and temporal taggers in NLP system are responsible for this process.

According to the TimeML annotation guidelines and Alfonso et al. work [Alonso and Strötgen, 2011], temporal expression includes:

- **offset**: the start and end position of the expression in the document
- **type**: whether the expression is of type date, time, duration, or set
- **value**: the normalized value of the expression

To identify this information, i.e., to extract and normalize temporal expressions, temporal taggers are applied after the text is preprocessed. Usually, sentence and token boundaries are detected and a part-of-speech tag is associated with every token. This information can then be used by the **temporal tagger** to identify temporal expressions.

As described above, context information has to be identified to determine the correct reference time and the temporal relation between a temporal expression and its reference time. While there are rule-based and machine learning based approaches for the extraction of temporal expressions, the normalization is usually done in a rule-based way by all temporal taggers as it has been used in this work.

In the machine learning approach, the classification problem can be described in the following way: For every token *t*, decide whether ***t*** is outside (O) of temporal expressions, inside (I) a temporal expression, or the beginning (B) of a temporal expression. The well-known IOB-format can be used for annotating tokens according to their property.

In the rule-based approaches to extract temporal expressions, they are usually based on regular expressions although they may use other information about the text as well, such as part-of-speech information. The more difficult task is the normalization of the temporal expressions. While explicit expressions can be normalized without further knowledge, the normalization of relative expressions is challenging.

1.4 Applications and Research Trends

Time clearly plays a central role in any information space, and it has been studied in several areas like information extraction, topic-detection, question-answering, query log analysis, and summarization. Time and temporal expressions can help recreating a particular historical period or describing the chronological context of a document or a collection of documents. Time can be in particular valuable for exploring search results along well-defined timelines and at multiple time granularities due to the key characteristics of temporal information described by alonso et al [Alonso and Strötgen, 2011]:

- **Temporal information is well-defined**: temporal expressions are clear in the sentence and has their own characteristics. They point to specific time or to an interval of time and they can be connecting to each other by some words like “past”, “Before”, “Next”.
- **Temporal information can be normalized:** Regardless of the used terms or the used language, every temporal expression referring to the same semantics can be normalized to the same value in some standard format. This property makes temporal information term and language-independent.
- **Temporal information can be organized hierarchically:** Temporal expressions can be described in different dimension e.g., of type day (“May 20, 2105”) or of type year (“2015”). Due to the fact that years consist of months, and months and weeks consist of days, temporal expressions can be mapped to coarser granularities based on the hierarchy of temporal expressions.

Using these key characteristics, temporal information about documents can be used to develop time-specific information retrieval and exploration applications. Now, we outline a number of applications that can benefit from leveraging more temporal information either by temporal expressions or timestamps. For each application, we describe why it is important and present a number of challenges.

Alfonso et al did an enrich research in the temporal information retrieval applications which are listed here:

1.4.1 Explanatory Search

Research in exploratory search systems has gained a lot of attention lately as they add a significant user interface component to help users search, navigate, and discover new facts and relationships. As the amount of information on the Web keeps growing, exploratory search interfaces are starting to surface. That said, it is not clear how to leverage temporal information. A few problems are:

- How to expose temporal information in exploratory search systems?
- What’s the best way of presenting temporal information as a retrieval cue?
- For which data sources, besides news, does exploratory search make sense?
- Is e-discovery a vertical application that can benefit from temporal information?

1.4.2 Micro-blogging and Real-time Search

Micro-blogging sites like Twitter have gained a lot of attention lately as the ultimate mechanism to broadcast what’s going on. Due to its nature, a typical message is very short and its lifespan is basically the crowd interest about that particular event be a football final game

or an earthquake. In the case of Twitter, it is very difficult to beat the timely broadcasting of an important event if one compares this to a news article. Each tweet has a timestamp but the organization of such information is still not clear. In the news context, the reporter has to write an article that contains a few paragraphs and submit the final version through some content management version that would push it to an external website so a search engine can hopefully crawl and index it in time. In parallel, if a tweet is so important by the time the reporter is finishing with the article, the main idea would be trending in Twitter, therefore highlighting its importance at a world scale. This is very similar to the traditional notion of topic detection and tracking, with one key difference: speed to detect that the topic is important and therefore a candidate for trending. Some problems:

- What is the best way to provide a timeline of events in micro-blogging?
- What is the lifespan of the main event?
- How fast and precise can one detect trending events?
- What is the fraction of new content on the topic stream?

1.4.3 Temporal Summaries

There has been seminal work on temporal summaries of news topics that shows how important temporal information is. One extension is to generate time sensitive summaries that can be used as temporal snippets. By design, the main goal of a snippet (or caption) is to present a document surrogate that the user can quickly scan in the search results page without the need to click and read the full content of a document. There is a limit to the number of lines of text that the snippet should present. Interesting questions include:

- When to show a timestamp or temporal expressions?
- Should the snippet present the matching lines in a timeline?
- Is a temporal summary a good surrogate for a document?
- For which kind of queries is a temporal summary appropriate?
- Should temporal summaries be query independent?

1.4.4 Temporal Clustering

The temporal information extracted from documents can directly be used to allow the user of a search engine to constrain his/her query in a temporal manner. That is, in addition to a textual part, a query contains a temporal part. For example, in addition to “world war” a temporal constraint like “1944-1945” could be specified. The user would obviously expect documents about World War II as results for his query. The objective when using a combination of a text and a temporal query can thus be formulated in the following way: The more both parts of the query are satisfied, i.e, the more the textual and the temporal parts fit to a document, the higher should be the rank of this document. The main problems for such a combination of constraints are the following:

- How can a combined score for the textual part and the temporal part of a query be calculated in a reasonable way?
- Should a document in which the “textual match” and the “temporal match” are far away from each other be penalized?

- What about documents satisfying one of the constraints but “slightly” fail to satisfy the other constraint?

1.4.5 Temporal Question Answering

To be able to answer time-related questions, a question answering system has to know when specific events took place. For this, temporal information can be associated with extracted facts from text documents. While this may be applicable for famous facts and events, question answering systems are often faced with imperfect temporal information. For this, identifying relationships between events described in documents is important as it is for many further NLP tasks Especially historic events tend to have a gradual beginning and ending so that knowing the temporal relationship between two events may allow answering a temporal query although no explicit temporal information is associated with the events. Research issues are:

- How can inconsistent temporal information be dealt with?
- How can temporal reasoning be executed if temporal relationships are missing?

1.4.6 Temporal Similarity

A related research question to temporal querying is temporal document similarity. Instead of comparing a temporal query with the temporal information of a document, two documents can be compared with respect to their temporal similarity. The main problem arising here is what makes two documents temporally similar? This leads to the following questions:

- Should two documents be considered similar if they cover the same temporal interval?
- Should the temporal focus of the documents be important for their temporal similarity?
- Can two documents be regarded as temporally similar if one contains a small temporal interval of the other document in a detailed way?

1.4.7 Timelines and User Interfaces

One important use of time entities of a document is to create a sorted list of events, a timeline. A timeline can be shown as a list of vertical textual items or visualized in many different ways. For example, as in Yahoo!’s Correlator. More sophisticated visualizations allow to focus on specific named entities with respect to time like in Yahoo!’s News Explorer. Here, interesting questions are:

- What is the appropriate way to present a timeline?
- Is a linear timeline the only way to present and anchor documents in time?
- How can one leverage document temporal measures to present a good display?
- Are there specific visualizations or user interfaces that can benefit from temporal information?

1.4.8 Searching in Time

Time entities can also be used to search in documents or log files that can be used to search the past for different purposes such as digital forensics, historical analysis or linguistic analysis. We can even search the future, for example, in news for events that are scheduled or may happen in the future. This idea is supported in the Yahoo! News Explorer tool already

mentioned. Microsoft Academic Search is an example of presenting publications and citations in a timeline. Some problems are:

- Besides news, what other sources would one like to use to search in the past and/or the future?
- How far does one need to go back in time?
- Can we improve bibliographic search instead of just sorting by publication date?
- How can we evaluate the quality of such systems?

1.4.9 Web Archiving

The goal of Web archiving is to collect and store digital content so that it is accessible for future tasks. Besides the detection of spam, which can be dealt with analyzing the evolution of the link structure of web pages, a main challenge in Web archiving is to take care of the temporal coherence of Web pages since it is not possible to collect all pages at the same time. Thus, the content of parts of the collection may change during the crawling process. Introduce a coherence framework to overcome the temporal diffusion of the Web crawls, i.e., to minimize the risk of in coherences. Nevertheless, open problems remain:

- How can temporal information are used to predict which pages are likely to change over time?
- How can temporal coherence be achieved for any point in time or time interval?

1.4.10 Spatio-temporal Information Exploration

Recently, there has been some research on combining spatial and temporal information extracted from documents for exploration tasks. In the same way as temporal information can be normalized using a timeline, spatial information can be normalized according its latitude/longitude information. To extract geographic expressions from documents, so-called geo taggers can be applied. Combining the information extracted from a temporal tagger with the information extracted from a geo tagger allows the exploration of documents according to the events mentioned in the text since events usually happen at some specific time and place. A system for the exploration of such spatio-temporal information from documents is TimeTrails. Some questions are:

- What's the best way to represent maps and time?
- Which kinds of scenarios can benefit from spatio-temporal exploration?

So if we could have a system which could recognize and tag these temporal expressions and convert the events related to these temporal expressions and convert all of these relationships into the graph which could be process-able for another machines, then many of these application could come to the reality and business one day.

1.5 Scope and Challenges

The problem of knowledge extraction from text is still only partly solved; this is particularly true if we consider it as a means for populating the Semantic Web. Being able to automatically and fastly produce quality linked data and ontologies from an accurate and reasonably complete analysis of natural language text would be a breakthrough: it would enable the development of applications that automatically produce machine-readable information from Web content as soon as it is edited and published by generic Web users. Existing Ontology Learning and Population (OL&P) approaches can be used as drafting ontology engineering tools, but they show some limitations when used to produce linked data on the Web:

- They usually need a training phase, which can take a long time
- Their output form needs further, non-trivial elaboration to be put in a logical form
- They only partially exploit OWL expressivity, since they typically focus on specific aspects e.g. taxonomy creation, disjointness axioms, etc.
- In most cases, they lack implementation of ontology design good practices
- Linking to existing linked data vocabularies and datasets is usually left as a separate task or not considered at all.
- Time and temporal dimension have not been considered strongly in contextual linked data

In other words, existing tools focus mainly on helping users identifying key elements for ontology drafting, assuming that they would transform and substantially refine and enrich the output. Our approach instead focuses on producing ontologies and linked data ready for the Web and creates knowledge data storage with focusing on temporal dimension of context.

In this research work scope, we focused on English contents and retrieving the English temporal expressions.

1.6 Goals and Contribution

We present a novel approach implemented in an online tool named AETAS, which performs deep parsing of natural language, extracts relations based on temporal dimension of context, and produces linked data graphs.

An important aspect of OL&P is the design quality of the resulting ontology: this is related to representing the results in a logical form such as OWL by ensuring that some best-practices are followed. Based on this consideration, we summarize a set of requirements that AETAS follows in order to enable robust OL&P:

- Ability to capture accurate semantic structures
- Ability to capture temporal expression form context
- Representing complex relations
- Enabling Temporal dimension to ontology graph
- No need of large-size domain-specific text corpora and training sessions
- Minimal time of computation
- Store the OL&P in the RDF graph database
- Ability to visualize the events based in Timeline representation

1.7 Work Plan

In order to implement the project aims as defined above, then the work needed to be done can be broken down into a number of components. The first is to import the SUTIME temporal tagger library into the system and plug Xlike project with the system via restful API, the second is to investigate and implement a mapper and RDFizer component into the system, and the final is to evaluate the system performance regarding to current dataset.

Below, more in-depth plans for the system are outlined, followed by a Gantt chart of the identified tasks and estimated workload. The estimated workload errs on the side of caution in order to allow for slippages, and additionally, weekends are not included in the time planning which allows for additional slack to be utilized if needed.

A final point to be made is that of the actual writing of the dissertation. There is no single "write up" task identified below, instead the actual writing of the dissertation is incorporated into the various tasks, but the final organizing in writing the current thesis has been considered in the time schedule.

1.7.1 Reusing Temporal tagger

In order to keep with good software engineering practices and follow the SOA framework, the SUTIME tagger library will be reused in the core part of AETAS. The first task required for this system will be to set up a library and an appropriate continuous integrating with the system for to be used in the whole system. Following this, work on the library and reuse it .NET C# and test them in order to ensure about its quality.

Once the library has been imported, the rule sets for both the recognition and normalization components need to be defined and imported in the system, which will be done by using the rules in the Stanford SUTIME.

ID	Task Name	Estimated Workload
1	Investigating on Temporal Information Retrieval	4 week
2	Investigating on temporal tagger and their evaluation	2 week
3	Importing the SUTIME library	3 days
4	Implementing the SUTIME consumption classes	2 weeks
5	Adding a front end to the system	1 weeks

1.7.2 Connecting to Linguistic semanticizer

Then AETAS need to be connected with WP2 Freeling Xlike web service which is in charge of natural language processing and ontology creation out of text. AETAS connect to Xlike through restful API. All requests send via HTTP port and the response would be in ZIP file contain the result of linguistic processes and time consumed for the process. The results are in JSON format. Following this, AETAS require well integration with XLIKE to avoid any breaks in the system.

ID	Task Name	Estimated Workload
1	Negotiating with TALP UPC research group	3 Days
2	Implementation of Service consumption	2 week
3	Enhance and test the service connectivity	3 days

1.7.3 Implement internal components

Following the implementation of the internal component which called Mapper and RDFizer, an investigation into RDFizer based techniques that could be implemented as modules for the system. This work will be experimental and implemented building as an inspiration for further work.

Accuracy and well match of mapper component with Xlike and temporal tagger are most important components that require lots of time and efforts. RDFizer would be very important in the case that should follow all rules and syntaxes that the graph storage database needs to be supported.

Running the Graph database (brightstardb) and integrat with the system would be another part of project that needs to be investigating on how highest performance can be used to enhance the system quality.

ID	Task Name	Estimated Workload
1	Implementation of phase 1 of Mapper	2 weeks
2	Implementation of phase 2 of Mapper	2 weeks
3	Test and Evaluate the mapper	3 days
4	Investigating on RDF graph systems	2 weeks
5	Investigating on RDFizing and its rules	2 weeks
6	Implementation of RDFizer and connect it to database	1 week

1.7.4 Information retrieving and visualization

Creating Layer for the user to be able to query over the graph database and retrieve the information in RDF format is another part of the project. In AETAS instead of normal SAPRQL, T-SPARQL has been used to let the system working well with graphs with focus on temporal aspect. Since our graph database would not support t-SPARQL, the mapper has been made in order to translate the t-SPARQL syntaxes into SPARQL standard syntaxes. And finally show them in visualized view to the user.

ID	Task Name	Estimated Workload
1	Investigating on SPARQL	3 Days
2	Implementation of t-SPARQL and Mapper	2 weeks
3	Investigating on Appropriate data visualizer	3 days
4	Connecting information retrieval layer to visualizer	2 weeks
5	Test and Evaluate the component accuracy	3 Days

1.7.5 Preparing Dataset and Testing the System

In order to evaluate the final tool, the system should work with proper and real data. Accordingly for this project, a quite huge raw content has been considered. The dataset should have enough reasonable temporal expressions and should be in the way that their content would be testable after processing. So the news articles would be good candidate for this purpose.

Then in order to see how temporal expressions are important, the results should be carefully processed and see how are they can be useful.

ID	Task Name	Estimated Workload
1	Preparing the Dataset	1 week
2	Cleaning and manipulating the dataset	2 Days
3	Run the system on the dataset	1 day
4	Checking the results and evaluate the system	1 week

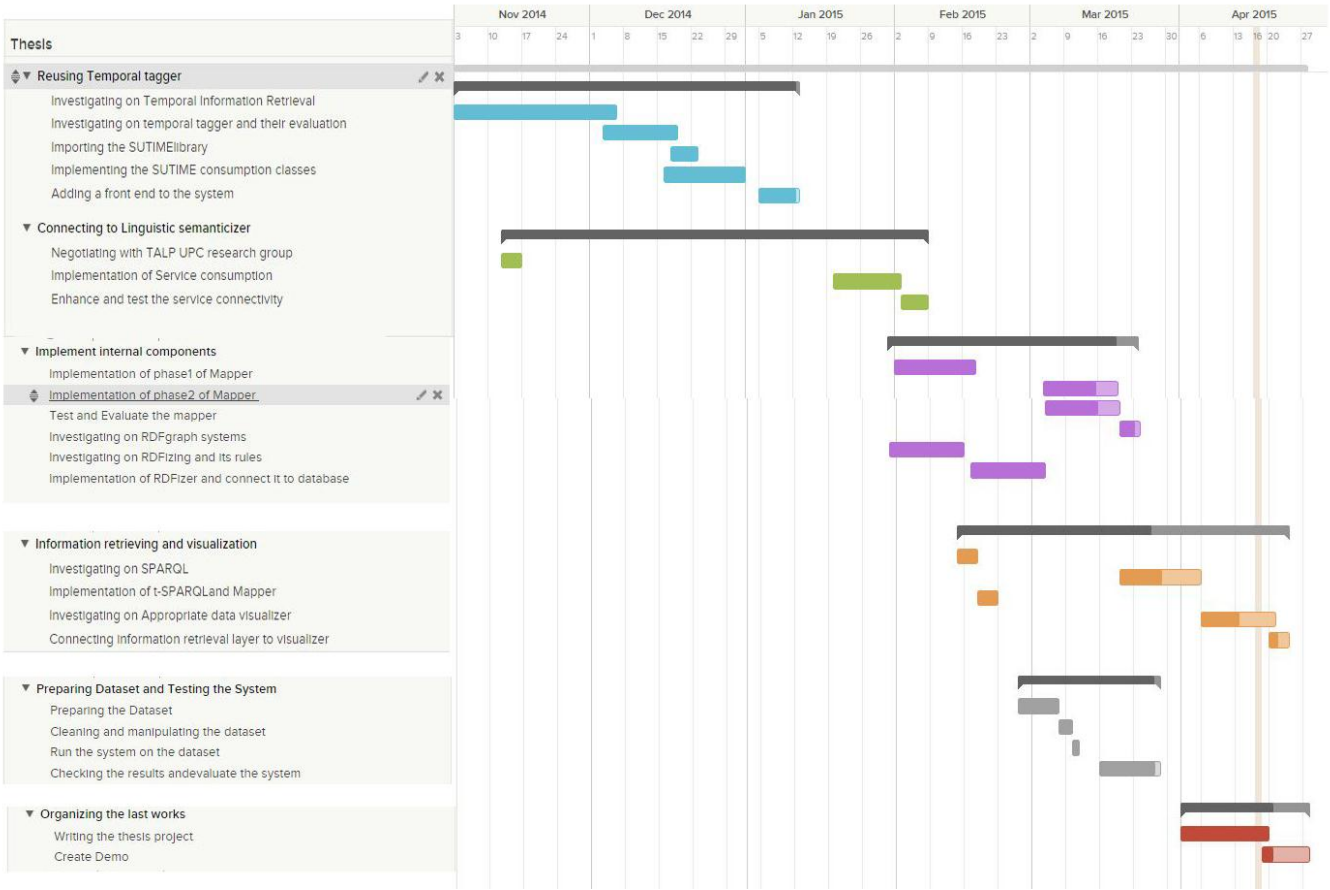

Figure 2: Gantt Chart of Project

II. Related Work

Temporal processing of text documents in terms of the extraction and normalization of temporal expressions as well as the extraction of temporal relations between events is very important for several NLP tasks requiring a deep understanding of language such as question answering or document summarization. Due to this fact, there has been significant research in temporal annotation of text documents. The markup language TimeML has become an ISO standard for temporal annotation [Pustejovsky et al., 2003], and the TimeBank corpus was developed [Pustejovsky, Hawks et al., 2003]. The latest version of the TimeBank corpus contains 183 news articles and can be regarded as the gold standard for temporal annotation. However, there has been important research activity before, and several evaluation challenges have been held to bring forward research in the area of temporal information extraction as described below.

Research work on fully utilizing the temporal information embedded in the text of documents for exploration and search purposes is very recent. The work by Alonso et al. [Alonso and Strötgen, 2011] presents an approach for extracting temporal information and how it can be used for clustering search results. [Shilder et al, 2001] presented semantic tagging for temporal expressions on News articles. They used their system to be part of part of an experimental multi-document summarization system while covers indexical and vague temporal expressions. Meanwhile an interesting visualizing system proposed by [Marcus et al. ,2011] their system extract and visualize the events from micro-blogs and tweets which are already being used for social science and augmented media experiences. FRED, RDFizer system from natural language text has been proposed by [Draicchio et al., 2013]. They represent the natural language into RDF framework which is connected to DBPedia knowledge source but without considering the temporal expressions.

Also some work has been done in order to combine documents into RDF and linked data approach and also with consideration of temporal expressions approach [Batsakis et al., 2011]. Also very interesting work has been by [Rula et al., 2012]. They investigated on availability of temporal information on Linked Open Data world among Billion Triple Challenge 2011 dataset. They found out availability of temporal information describing the history and the temporal validity of statements and graphs is still very limited. Meanwhile [Batsakis et al., 2011] proposed new framework to handle Spatial-Temporal information in OWL 2.0.

To the best of our knowledge, there is still no formal study of temporality issues in RDF graphs and RDF query languages. Temporal database management has been extensively studied, including data models, mostly based on the relational model and query languages.

III. SYSTEM Architecture

AETAS as a Latin word with meaning Time and Era, is the name of our system. AETAS is fully designed and implemented in Service oriented approach in order to get the benefits of SOA systems. AETAS has four main components which are responsible for different functionalities. Two components have been implemented in-house: Temporal Mapper and RDFizer. The other two components are SUTIME and FreeLing which are integrated as external components. In this section, first of all, the overview of AETAS architecture has been explained how they are connected to each other and then we go into detail of each component to see how they work and why they have been chosen.

6.1 AETAS Overview

The process starts with a raw text document with their reference date, obtained from some web site such as news agencies or blogs. The document is fed to Temporal tagger component (SUTIME in our system), which returns annotated expressions. Then the Mapper component receives the document, and after some preprocessing, runs the document through FreeLing language processing web-service. Then, Mapper combines the linguistic information provided by FreeLing and the temporal expressions identified by temporal tagger. The output of Mapper is piped to RDFizer, which creates triples and linked information based on the semantic representation made by FreeLing, and stores them into a BrightStarDB1 native RDF database.

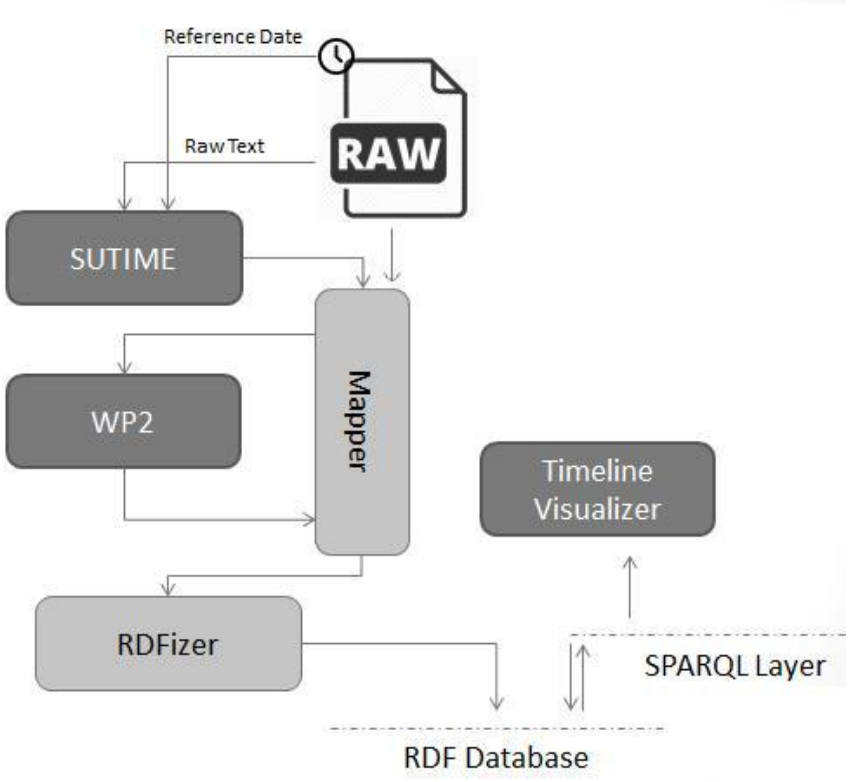

Figure 3: AETAS Architecture

Then another tool has been provided to fetch the data and import them in the visualizer component.

As has been said above, AETAS has external components that connect them via the Restful service API to the web services and received processed information.

In the section below the SUTIME as a temporal tagger has been described and also the reasons why this system has been selected among other has been described.

6.2 Temporal Tagger

Temporal expressions are recognized as being highly idiosyncratic, at least in English, but attempts have been made by linguists to make generalizations of the underlying grammar. Rule-based automated annotators use this principle by attempting to annotate timexes using these rule-based generalizations of the grammar.

Many efforts and researches has been done in temporal expression applications which some of them has been explained and the reason why SUTIME has been selected among them.

1.7.1 TempEx and GUTime

TempEx (Mani & Wilson, 2000) is a rule-based tagger that accepts a document tokenized into words and sentences and tagged for part-of-speech. A number of regular expression rules are used to define the extent of what should be tagged.

TempEx deals with the normalization of self-contained expressions, and then “discourse processing module”, deals with relative expressions. For relative times, a reference time is established, either from the context of the surrounding sentences or the document creation date, and then rules handle temporal expressions representing offsets from this date by first computing the magnitude of the offset (e.g., “month”, “week”, etc), and then the direction, either from direct indicators (e.g., “last Thursday”) or from the tense of the sentence.

GUTime (Verhagen, et al., 2005) is an extension to the TempEx tagger that extends the capabilities of TempEx to include the new TIMEX3 tag defined in TimeML, as well as some expressions not handled by TempEx, such as durations, some temporal modifiers, and European date formats.

This program structure makes adding or changing rules difficult due to the coupling between the rules and the logic itself, and makes analysis of the rules difficult.

1.7.2 Chronos

Chronos (Negri & Marseglia, 2004) was a system created for the 2004 TERN evaluation that, like GUTime, provides one system for recognition and normalisation. However, these two tasks are split into separate internal components. Chronos is designed to be a multi-lingual system, coping with both English and Italian text.

One main difference between Chronos and GUTime is that Chronos can handle plaintext; tokenisation and part-of-speech tagging occurs in the first phase of the program. This does have the downside of making Chronos more difficult to componentize; if it were to be incorporated into a larger system, this pre-processing may want to be separated out to use a better system.

Additionally, Chronos, in contrast to GUTime which has a clear separation of components, appears to have a heavier coupling and a more integrated system. This recognition phase results in an intermediate representation – an extension of the TIMEX2 standard – which provides the metadata detected in the recognition phase as additional attributes to a tag.

Normalisation continues in a similar way to that proposed by (Mani & Wilson, 2000). Expressions are classified as either being absolute or relative, and then in the case of relative dates, the direction and magnitude of the relativity is determined and combined with a base date (determined in the recognition phase) to produce an anchor in time.

1.7.3 HeidelTime

Another temporal tagger which has been introduced in SemEval 2010 is HeidelTime. HeidelTime is a rule-based system mainly using regular expression patterns for the extraction of temporal expressions and knowledge resources as well as linguistic clues for their normalization. In the TempEval-2 challenge, HeidelTime achieved the highest FScore (86%) for the extraction and the best results in assigning the correct value attribute, i.e., in understanding the semantics of the temporal expressions.

HeidelTime 2.0 achieved high quality results for the extraction and normalization of temporal expressions. The precision-optimized rule set achieved the best results for interpreting the semantics of the temporal expressions. Assigning the correct value attribute, is crucial since the value is used for further analysis of the documents, e.g., when ordering events or doing a temporal analysis of documents. The rule-based approach makes it possible to include further knowledge easily, e.g., to assign temporal information directly to historic events.

1.7.4 SUTIME

SUTIME (Angel. X Chang and Christopher, 2012) is a temporal tagger system for recognizing and normalizing temporal expressions in English text. SUTIME is available as part of the Stanford CoreNLP pipeline and can be used to annotate documents with temporal information. It is a deterministic rule-based system designed for extensibility. SUTIME as a Java library, implemented as an annotator in the Stanford CoreNLP pipeline; its main features are described below:

1. **Extraction of temporal expressions from text**: Given tokenized English text, SUTIME finds temporal expressions and outputs annotations for further manipulation and interpretation. Its output includes annotations in the form of TIMEX3 tags. TIMEX3 is part of the TimeML annotation language for markup of events, times and their temporal relations in documents.
2. **Representation of temporal objects as Java classes**: Natural language uses many kinds of temporal expressions. SUTIME provides tools to map them to logical representations and data structures that are easier to handle programmatically.
3. **Resolution of temporal expressions with respect to a reference date**: When processing natural language text, one often has to work with expressions that refer to a relative time (e.g., last Friday). Determining the actual date to which such expressions refer requires a reference date, on which the statement was made. SUTIME uses document dates as references.

SUTIME is one of best currently available temporal taggers, as has been shown on TempEval-2 evaluation on 2010. Other top-performing systems are GUTIME, HeidelTime and TRIPS/TRIOS. Based on SUTIME developer and author and Experimental results show that both the rule-based system (HeidelTime) and the probabilistic system (TRIPS/TRIOS) were as effective (with similar F1 scores) for identifying temporal expressions. This validates our intuition that most temporal patterns can be captured effectively with rules. SUTIME has the highest overall F1 and the highest recall in discovering temporal expressions. Compared to Heidel-Time2, the high recall system, SUTIME has both higher precision and higher accuracy for the attribute type and value. However, compared to HeidelTime1, SUTIME has a lower precision and lower accuracy for the attribute type and value. Note that using the attribute scores of the official TempEval-2 scorer to compare the systems can be misleading since the attributes are scored only for tokens correctly identified as belonging to a temporal expression. Because the attribute scores are computed using a total that is different for each system, it is inappropriate to compare the attribute scores across systems. For example, consider a system that only marks one token as belonging to a temporal expression. Assuming that the token had the correct type and attribute, the system would achieve an accuracy of 100% on the type and value attributes (although the F1 of identified extents would be extremely low). From this example, we see that is it difficult to only compare the attribute scores without considering the scores for identifying extents.

In the figure below, the rank of SUTIME in compare with other has been shown:

	Extents			*Attribute*	
System	P	R	F_1	type	value
GUTime	0.89	0.79	0.84	0.95	0.68
SUTime	0.88	**0.96**	**0.92**	0.96	0.82
TRIPS/TRIOS	0.85	0.85	0.85	0.94	0.76
HeidelTime1	**0.90**	0.82	0.86	**0.96**	**0.85**
HeidelTime2	0.82	0.91	0.86	0.92	0.77
HeidelTime*	0.57	0.89	0.70	0.96	0.85

Figure 4: Temporal Tagger Comparison

Given its performance and ease of use, we decided to integrate SUTIME in our system. We use SUTIME for its three main features: (1) Extraction of temporal expressions from text, (2) Representation of temporal expressions as objects convenient to handle programmatically, and (3) Resolution of temporal expressions with respect to a reference date (e.g. the document date). The very simple flow of SUTIME has been figured in Figure 2. These results are most precious part of AETAS which would be used in all further components.

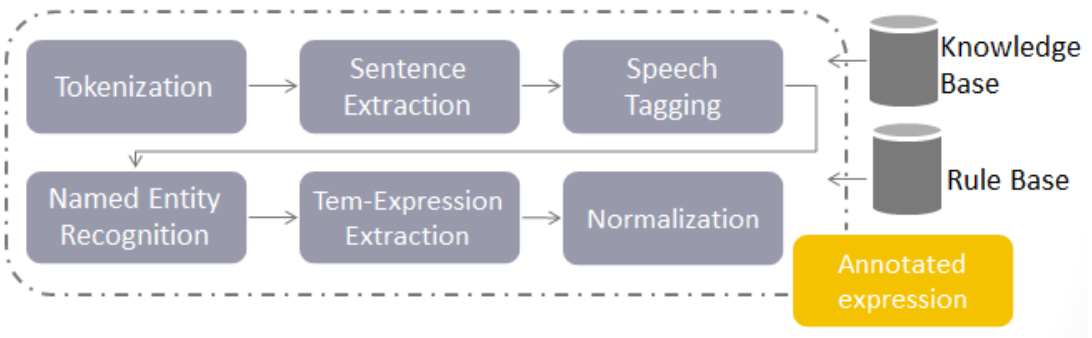

Figure 5: SUTIME Architecture

According to SUTIME architecture, Knowledge Base and Rule Base are external sources that can imported into the system which means they can be updated manually or automatically without changing the code.

So after importing the resources into the system and input the raw text, SUTIME tokenize the linguistic features and extract the Sentences, then recognize the entities, and pull out the temporal expressions. The final step would be normalizing the expressions and return them as object class to the program. AETAS use the objects and send it to the mapper component for further processes.

6.3 WP2: Language Processor Pipeline:

WP2 is devoted to building language analysis pipelines that will extract from texts the core knowledge that the project is built upon. The different language functionalities are implemented following the service oriented architecture (SOA) approach defined in the European project XLike.

The project has been done by collaboration of are Jo˘zef Stefan Institute (JSI), Karlsruhe Institute of Technology (KIT), Universitat Polit`ecnica de Catalunya (UPC), University of Zagreb (UZG), and Tsinghua University (THU) and The Spanish company iSOCO is in charge of integration of all components developed in the project.

The goal of the XLike project is to develop technology which enables gathering documents in a variety of languages and genres (news, blogs, tweets, etc.) and extracting language-independent knowledge from them, in order to provide new and better services to publishers, media monitoring and business intelligence.

Therefore all the pipelines (one for each language) have been implemented as web services and may be requested to produce different levels of analysis (e.g. Tokenization, lemmatization, NERC, parsing, relation extraction, etc.). This approach is very appealing due to the fact that it allows to treat every language independently and to execute the whole language analysis process at different threads or computers allowing an easier parallelization (e.g., using external high performance platforms such as Amazon Elastic Compute Cloud EC2 as needed. Furthermore, it also provides independent development life-cycles for each language which is

crucial in this type of research projects. Recall that these web services can be deployed locally or remotely, maintaining the option of using them in a stand-alone configuration.

Each language analysis service is able to process thousands of words per second when performing shallow analysis (up to NE recognition), and hundreds of words per second when producing the semantic representation based on full analysis. For instance, the average speed for analyzing an English document with shallow analysis (tokenizer, splitter, morphological analyzer, POS tagger, lemmatization, and NE detection and classification) is about 1,300 tokens/sec on a i7 3.4 Ghz processor (including communication overhead, XML parsing, etc.). This means that an average document (e.g, a news item of around 400 tokens) is analyzed in 0.3 seconds.

When using deep analysis (i.e., adding WSD, dependency parsing, and SRL to the previous steps), the speed drops to about 70 tokens/sec, thus an average document takes about 5.8 seconds to be analyzed. The parsing and SRL models are still in a prototype stage, and we expect to largely reduce the difference between shallow and deep analysis times.

However, it is worth noting that the web-service architecture enables the same server to run a different thread for each client without using much extra memory. This exploitation of multiprocessor capabilities allows a parallelism degree of as many request streams as available cores, yielding an actually much higher average speed when large collections must be processed.

1.7.5 Semantic Representation

Apart from the basic state-of-the-art tokenizers, lemmatizers, PoS/MSD taggers, and NE recognizers, each pipeline requires deeper processors able to build the target language-independent semantic representation. For that, we rely on three steps: dependency parsing, semantic role labeling and word sense disambiguation. These three processes, combined with multilingual ontological resources such as different WordNets, are the key to the construction of our semantic representation.

1.7.6 Dependency Parsing

In XLike, we use the so-called graph-based methods for dependency parsing. In particular we use MSTParser for Chinese and Croatian, and Treeler –a library developed by the UPC team that implements several methods for dependency parsing, among other statistical methods for tagging and parsing– for the other languages.

1.7.7 Semantic Role Labeling

As with syntactic parsing, we are using the Treeler library to develop machine-learning based SRL methods. In order to train models for this task, we use the treebanks made available by the CoNLL-2009 shared task, which provided data annotated with predicate-argument relations for English, Spanish, Catalan, German and Chinese. No treebank annotated with semantic roles exists for Slovene or Croatian yet, thus, no SRL module is available for these languages in XLike pipelines.

1.7.8 Word Sense Disambiguation

The used Word Sense Disambiguation engine is the UKB implementation provided by FreeLing. UKB is a non-supervised algorithm based on PageRank over a semantic graph such as WordNet.

Word sense disambiguation is performed for all languages for which a WordNet is publicly available. This includes all languages in the project except Chinese.

The goal of WSD is to map specific languages to a common semantic space, in this case, WN synsets. Thanks to existing connections between WN and other resources, SUMO and OpenCYC sense codes are also output when available. Finally, we use PredicateMatrix –a lexical semantics resource combining WordNet, FrameNet, PropBank, and VerbNet– to project the obtained concepts to PropBank predicates and FrameNet diathesis structures, achieving a normalization of the semantic roles produced by the SRL (which are treebank-dependent, and thus, not the same for all languages).

1.7.9 Frame Extraction

The final step is to convert all the gathered linguistic information into a semantic representation. Our method is based on the notion of frame: a semantic frame is a schematic representation of a situation involving various participants. In a frame, each participant plays a role. There is a direct correspondence between roles in a frame and semantic roles; namely, frames correspond to predicates, and participants correspond to the arguments of the predicate. We distinguish three types of participants: entities, words, and frames.

It is important to note that frames are a more general representation than SVO-triples. While SVO-triples represent binary relations between two participants, frames can represent any n-ary relation. For example, the frame for plan is a ternary relation because it includes a temporal modifier. It is also important to note that frames can naturally represent higher-order relations: the theme of the frame plan is itself a frame, namely make.

Finally, although frames are extracted at sentence level, the resulting graphs are aggregated in a single semantic graph representing the whole document via a very simple co-reference resolution method based on detecting named entity aliases and repetitions of common nouns.

So as summary, WP2 as language analyzer is responsible for tokenizing, lemmatizing, PoS/MSD tagging and NE recognizing. And apart from these main functionalities as it figured in Figure 3, each pipeline requires deeper processors able to build the target language-independent semantic representation on three steps: dependency parsing, semantic role labeling and word sense disambiguation. These three processes, combined with multilingual ontological resources such as different WordNets, are the key to the construction of the semantic representation.

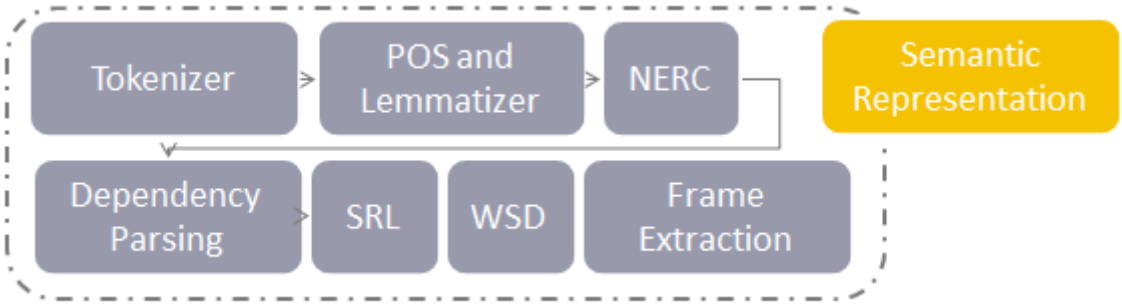

Figure 6: XLIKE Architecture

However Xlike as web-service is considered as black-box and we just call the service and post reformatted and processed raw text paragraph which has been prepared by first part of Mapper component as input and the results are semantic representation of sentences and events in JSON format. Then the second part of Mapper component will come into action and all temporal expressions that have been identified by SUTIME, will be mapped in appropriate place of content. The full functionality of Mapper component has been explained in section 3.

6.4 Temporal mapper

Temporal Mapper component has been used in the middle of AETAS process. This component has two parts. First part takes the temporal expressions from SUTIME from one side and takes the raw content text from another side. For Example "FC Barcelona has a plays *Tomorrow and next Monday from 8:00 PM to 10:30 PM* " sentence has three temporal expression as it shown in figure 4 in case reference date has been considered as 19th March 2015. The first two have TIME type and third one has Duration type. Temporal mapper change the text to "FC Barcelona has a play on *DAY 1* and *DAY 2 DAY 3*" . The reason of this replacement is because of WP2 component is not able to identify these expressions as temporal expression and process them as "noun" for *tomorrow* or 8:00 PM as different token as it is shown in figure 5.1. So in order to avoid this issue and also keep the linguistic reasoning of sentence, the Day 1 or Day 2 has been replaced, because WP2 consider them as Date token and represent the semantic approach in best appropriate way as it is shown in figure 5.2.

Text	Value	Timex3 Tag
tomorrow	2015-03-20	<TIMEX3 tid="t1" type="DATE" value="2015-03-20">tomorrow</TIMEX3>
next friday	2015-03-27	<TIMEX3 tid="t2" type="DATE" value="2015-03-27">next friday</TIMEX3>
from 8:00 PM to 10:30 PM	PT2H30M	<TIMEX3 beginPoint="t3" endPoint="t4" tid="t5" type="DURATION" value="PT2H30M">from 8:00 PM to 10:30 PM</TIMEX3>

Figure 7: SUTIME Temporal Tagging Result

```
  <conll>1  Fc_Barcelona fc_barcelona          NNP pos=noun|type=proper|neclass=person      B-PER 2  SBJ    _
2  has          have                 VBZ pos=verb|vform=personal|person=3          O     0  ROOT   00065639-v have.00
3  a            1                    CD  pos=number                               O     2  VC     _           _
4  play         play                 NN  pos=noun|num=s                           O     3  OBJ    07007945-n play.01
5  on           on                   IN  pos=preposition                         O     4  LOC    _           _
6  tommorow     tommorow             NN  pos=noun|num=s                           O     5  PMOD   _           _
7  and          and                  CC  pos=conjunction|type=coordinating       O     6  COORD  _           _
8  next_friday  [V:??/??/??:??.??:??] NNP pos=date                               O     7  CONJ   _           _
9  from         from                 IN  pos=preposition                         O     7  CONJ   _           _
10 8:00         [??:??/??/??:8.00:??] NNP pos=date                               O     9  PMOD   _           _
11 PM           pm                   NNP pos=noun|type=proper|neclass=organization B-ORG 9  PMOD   09907196-n  _
12 and          and                  CC  pos=conjunction|type=coordinating       O     9  COORD  _           _
13 10:00        [??:??/??/??:10.00:??] NNP pos=date                              O     12 CONJ   _           _
14 PM           pm                   NNP pos=noun|type=proper|neclass=organization B-ORG 12 CONJ  09907196-n  _
```

Figure 8: WP2 output without Mapper component

```
  <conll>1 Fc_Barcelona fc_barcelona          NNP pos=noun|type=proper|neclass=person B-PER 2 SBJ    _           _
2 has          have                 VBZ pos=verb|vform=personal|person=3    O     0 ROOT   00065639-v have.00 _ _
3 a            1                    CD  pos=number                          O     2 VC     _           _       _ _
4 play         play                 NN  pos=noun|num=s                      O     3 OBJ    07007945-n play.01 _ _
5 on           on                   IN  pos=preposition                     O     4 LOC    _           _       _ A1
6 DAY_1        [??:1/??/??:??.??:??] NNP pos=date                           O     5 PMOD   _           _       _ _
7 and          and                  CC  pos=conjunction|type=coordinating   O     5 COORD  _           _       _ _
8 DAY_2        [??:2/??/??:??.??:??] NNP pos=date                           O     7 CONJ   _           _       _ _
9 DAY_3        [??:3/??/??:??.??:??] NNP pos=date                           O     7 CONJ   _           _       _ _
```

Figure 9: Wp2 output after Mapper component

After calling the WP2 after preparing the raw text, the result has been sent back again to second part of Temporal mapper. The second part convert the JSON response into object approach and then tries to replace the temporal expressions with their appropriate value. The result after Temporal Mapper component is shown in figure 5.3.

```
1 FC_Barcelona fc_barcelona          NNP pos=noun|type=proper|neclass=organization B-ORG 2 SBJ    _           _       _ _
2 has          have                 VBZ pos=verb|vform=personal|person=3          O     0 ROOT   00065639-v have.00 _ _
3 a            1                    CD  pos=number                               O     2 VC     _           _       _ _
4 play         play                 NN  pos=noun|num=s                           O     3 OBJ    07007945-n play.01 _ _
5 on           on                   IN  pos=preposition                         O     4 LOC    _           _       _ A1
6 Tomorrow     [20/03/2015] NNP pos=date                                    O     5 PMOD   _           _       _ _
7 and          and                  CC  pos=conjunction|type=coordinating       O     5 COORD  _           _       _ _
8 Next_Friday        [23/03/2015] NNP pos=date                              O     7 CONJ   _           _       _ _
9 From_8:00_PM_to_10:00_PM       [P2H30M] NNP pos=date                              O     7 CONJ   _
```

Figure 10: Mapper phase 2 output

After mapping and processing the tokens which are objects at this stage, has been sent to RDFizer component.

6.5 RDFizer

The RDFizer component in AETAS is responsible for creating RDF triples based on semantic representation of context and temporal information.

Current approaches for coding temporal information consider it as additional data inside the data model. Therefore, temporal information is implicit in the data and difficult to access by programs. Accordingly, in AETAS system, time has been proposed as an additional semantic dimension of data. Therefore, it needs to be regarded as an element of the meta model instead of being just part of the data model.

By applying the foundations established by Gutierrez et al. (2007), implementing an RDF compatible syntax for temporal data is the first step. And second step is applying strategies for store these RDF triples. As in most of the approaches dealing with temporal entities, time would be modeled as a dimensional discrete value. More formally, each valid time point is defined as $\tau \in N_0$. A time interval consists of two time points s (for start) and e (for end) such that s, $e \in \tau | s \leq e$ and we can define durations type of temporal expressions. By allowing s to be equals e, we

are able to express time points type of temporal expressions. Since RDFizer tries to roll up the time dimension as much as possible and create another dimension in the graph, then connecting all possible same values in the node via "Associated with" edge is possible. This approach would increase the performance of information retrieval in SPARQL layer when the user try to drill down into temporal data. In the example mentioned above, the RDF graph representation of the sentences illustrated in figure 8. As it's shown, since From 8:30 Pm to 10:00 PM identified as Duration, the start time and end time has been considered. Expressions "Tomorrow" and "Next Monday" since refer to single point of time, their start time and end time has been calculated as equal and they just rolled up and another dimension has been created.

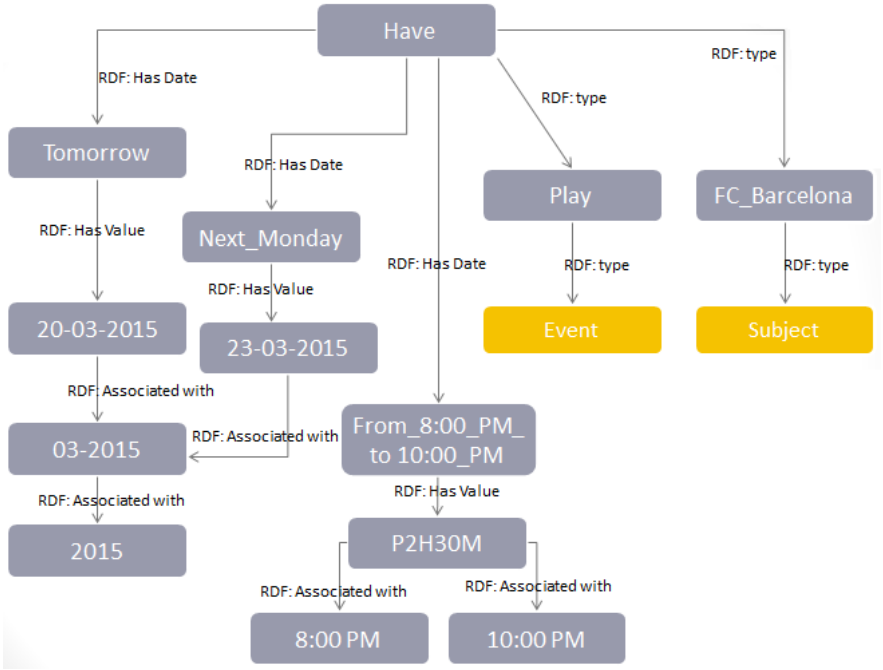

Figure 11: RDFizer Result Example

6.6 Information Retrieval: SPARQL Layer

As SPARQL layer is not part of AETAS system, but in order to access to stored data in the RDF database, a temporal extension of SPARQL called τ-SPARQL (Tappolet and Bernstein, 2009), has been considered. Then τ-SPARQL queries would automatically translated to standard SPARQL operating on storage scheme leveraging existing RDF/SPARQL infrastructure. Time point queries aim at retrieving information valid at a specified point in time. This is of special importance whenever a snapshot in the past needs to be retrieved from a dataset. The FROM SNAPSHOT τ expression signals the query engine to evaluate the query's graph pattern only on graphs-elements valid at the time point τ, where τ has to be a literal time value. For example, the query below retrieves all AETAS:EVENT that are valid in March, 2015.

```
Select ?Event FROM SNAPSHOT 03-2015 where {
      ?Event  a AETAS: Event
}
```

The biggest advantages of τ-SPARQL is, it can support time interval such as start date and end date. This means retrieving Events from two point of data which are store as a specific temporal point is possible:

```
Select ?s ?Event FROM SNAPSHOT where {
       [01-2015, 06-2015] ?Event  a AETAS: Event .
}
```

But since the RDF database can't support τ-SPARQL, a mapping method has been considered in order to map τ-SPARQL syntax into standard SPARQL syntax. A mapping has been implemented by following algorithm explained by Tappolet and Berstein (2009).

1.7.10 Mapping Time Point Queries

Rewriting a time point query requires three steps: First, the respective intervals for a time point needs to be determined. Second, the elements belonging to these intervals need to be combined to a new, virtual data set. Finally, the graph pattern needs to be matched against this new data set. As described above, the information about the validity of an interval is encoded in the default graph of the named graph set. To extract the intervals valid at a given time point, the partial query shown in Listing 5 has to be added to the initial query in the WHERE clause.

```
?g time: hasBeginning ?start .
 ?start time :[ inInteger|inXsdDateTime|...] ?s .
FILTER(?s <= <TP > || ?start = NOW) .
?g time: hasEnd ?end .
?end time :[ inInteger|inXsdDateTime|...] ?e .
FILTER(?e >= <TP > || ?end = EVER) .
```

The pattern in Listing above will bind variable **?g** to all the named graphs containing triples valid in time point <TP>. The expression [inInteger |inXsdDateTime | ...] is evaluated according to the literal type of <TP>. If it is a xsd:Integer, then the time:inInteger property is selected. Analogously, a xsd:dateTime will be represented by the inXsdDateTime property. Note that both the τ-SPARQL query language and the representation format are open to additional time formats as long as the rewriting algorithm is aware of the acceptable correct formats and their respective mappings. In a last step of the rewriting process, the initial graph pattern of the query needs to be restricted to the intervals bound in ?g. SPARQL offers this functionality with the GRAPH keyword. This is done by extracting the pattern inside the WHERE clause of the query and inserting it enclosed by GRAPH ?g {<pattern>}.

1.7.11 Mapping Temporal Queries

To map the queries that select the validity of a triple pattern and/or relations between validities, the rewriting process is slightly more complex. We define that all classic SPARQL variables ?v belong to a set C. The intervals [?s,?e] to belong to the set of intervals I, and ?s and ?e to belong to the set of point variables P. For each i ∈ I there exists exactly one URI g belonging to the set of URIs of named graphs G and a pair of time points ?s and ?e, which mark its start and end. Any pair of time points ?s and ?e can be mapped to an interval i, which in turn can be mapped to a URI from G. Consequently, a quad pattern in the form [?s,?e] <triple pattern> can be rewritten as GRAPH ?x{<triple pattern>} where ?x is the URI ∈ G that corresponds to the interval defined by [?s,?e]. Note, that a new variable "?x" needs to be chosen for every rewrite. Please note that temporal variables are not allowed in the predicate part.

```
<PREFIX time: http:// www.w3.org /2006/ time#>
<PREFIX foaf: http:// xmlns.com/foaf /0.1/#>
SELECT ?s2 ?e2 ?person WHERE{
GRAPH g1 { ?einstein foaf:name "Albert Einstein " .}
?g2 time: intervalOverlaps ?g1 .
GRAPH g2 { ?person a foaf:Person }.
?g2 time:hasBeginning ?start .
? start time:inInteger ?s2 .
?g2 time:hasEnd ?end .
?end time:inInteger ?e2 .
}
```

It is important that whenever a temporal variable is used outside the interval context (i.e., outside a GRAPH ?x{<triple pattern>}) then it needs to be treated like a time point statement. Hence, the partial query shown in Listing 5 needs to be added to the rewritten τ-SPARQL query. As an example, Listing 6 shows the rewritten standard SPARQL syntax of the query in Listing 4.

6.7 Information Visualization: TIMELINE

After the information relevant to the input query has been retrieved, it is sent to the TIMELINE component for presentation. TIMELINE generates an interactive graph based on Time-Event., allowing web site creators to embed interactive timelines into their sites.

The SIMILE Timeline web plugin allows adding interactive timeline components to the system. The plugin is open source software originally developed as part of the SIMILE project with funding from the Andrew W. Mellon Foundation.

Figure 5 shows an example of temporal visualization based on events on the timeline. User can click on the event and see the source refers to that event.

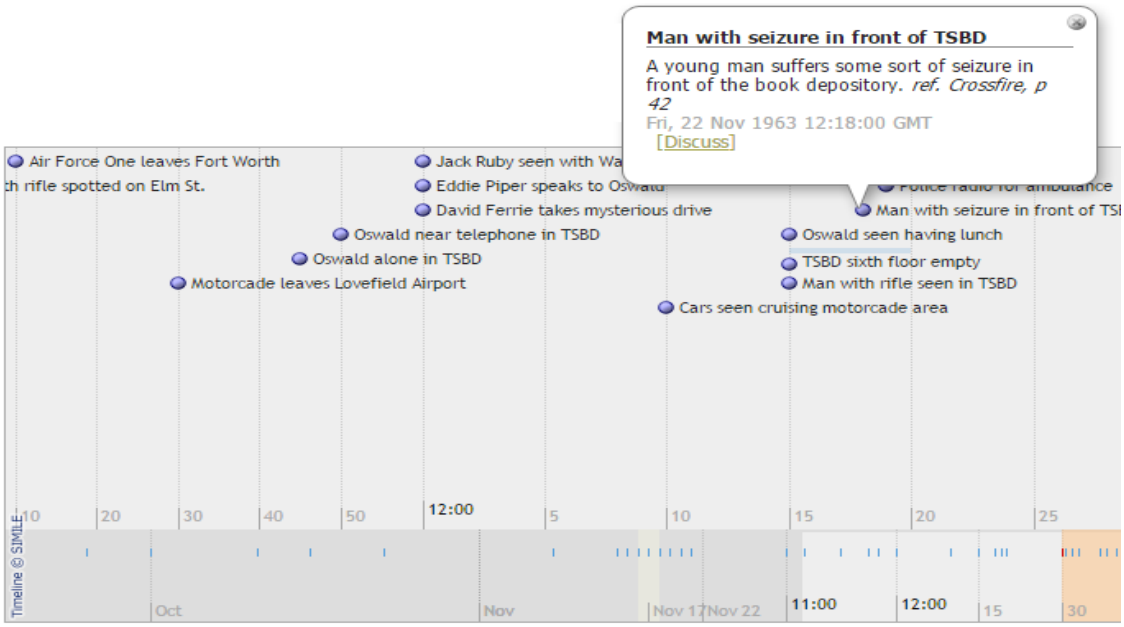

Figure 12: Visulization in Timeline

This is the example of how RDF retrieved by retriever component will be passed

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:foaf="http://xmlns.com/foaf/0.1/"
       xmlns="http://usefulinc.com/ns/doap#">
       <Project>
       <name xml:lang="en-US"
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Timeline</name>
       <homepage rdf:resource="http://simile.mit.edu/timeline/" />
       <created>2006-06-29</created>
       <shortdesc xml:lang="en-US"
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Timeline is a DHTML AJAX
timeline widget.</shortdesc>
       <description xml:lang="en-US"
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Timeline is a DHTML AJAX
timeline widget.</description>
       <mailing-list rdf:resource="http://simile.mit.edu/mail.html" />
       <maintainer>
              <foaf:Person>
                     <foaf:name xml:lang="en-US"
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David Huynh</foaf:name>
                     <foaf:homepage
rdf:resource="http://people.csail.mit.edu/dfhuynh/" />
                     <foaf:mbox rdf:resource="mailto:dfhuynh@mit.edu" />
              </foaf:Person>
       </maintainer>
       <!-- currently supported versions -->
       <release>
              <Version>
                     <created>2006-06-29</created>
                     <revision>1.0</revision>
```

```
                </Version>
        </release>
        <license rdf:resource="http://usefulinc.com/doap/licenses/bsd" />
        <repository>
                <SVNRepository>
                        <location
rdf:resource="http://simile.mit.edu/repository/timeline/" />
                        <browse rdf:resource="http://simile.mit.edu/viewsvn/timeline/"
/>
                </SVNRepository>
        </repository>
        <wiki rdf:resource="http://simile.mit.edu/wiki/" />
        <bug-database rdf:resource="http://simile.mit.edu/issues/" />
        <programming-language xml:lang="en-US"
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Javascript</programming-
language>
</Project>
</rdf:RDF>
```

In the next section, the Dataset and experiment would be described. In addition some statistical calculation has been done in order to show how temporal expressions are important in practice.

IV. AETAS Experiment

In order to test and evaluate our system, 1000 news pages from BBC news and New York Times has been collected. The collection has 26,698 paragraphs, 29,104 Sentences from 1986-01-24 to 2015-01-26. After running the system and reading and processing all dataset, AETAS found 8,690 temporal expressions representing 23% of the whole amount of paragraphs. Some statistical calculation has been done in order to see how temporal expressions behave in the context and how relevant they are. Figure 6 shows the result of a regression between temporal expressions and other elements in context. It shows how temporal expressions are relevant, and how when they are considered as a new information dimension, many more results can be retrieved.

```
Regression Equation

temp_count = 2,452 + 0,01499 word_count - 0,1346 sen_count + 0,0601 par_count
```

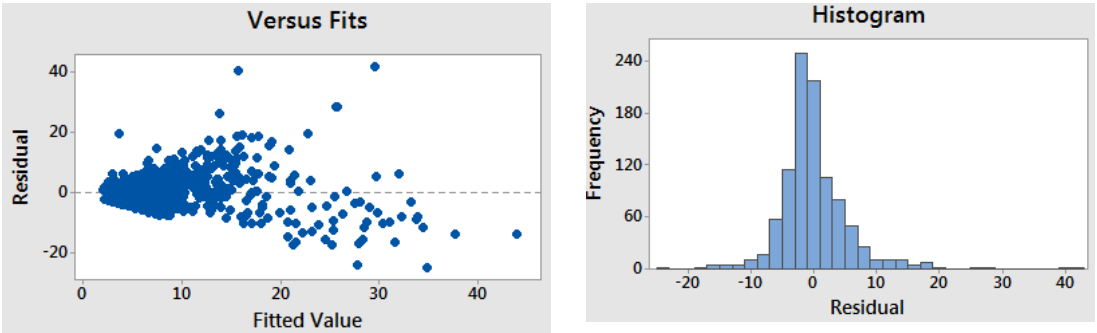

Figure 13: Regression calculation over dataset

According to regression calculated and is shown in figure ….. temporal has significant relationship with sentences in the context. But the negative coefficient shows if the number of sentences grows, it doesn't mean the number of temporal expressions would grow with same portion. In oractice, it means, in several sentences in the paragraph, just few sentences has temporal expressions and the rest of sentences refers that temporal point or interval in the content. But the positive coefficient means, by growing number of paragraphs, the temporal expressions grows and we can have at least one temporal expression by the regression equation shown above.

After running the system from our dataset, we found out 6,335 of the 8,690 extracted temporal expressions refer to exact time and have TIME data type, 1,119 are DURATION, and the remaining 1,236 are distributed among INTERVAL, SET, and AMBIGUOUS classes (varying dates such as Easter or Mother's Day).

In current stage, the system discards AMBIGUOUS expressions, but in the future, based on learning coming from stored triples; it can be added to the temporal rules and calculated by the system.

After analyzing the stored triples in database, we have 127 temporal expressions that after normalizing, they are referring to same point of time in day level and after rolling up, 439 of them pointing to same month and 1149 of temporal expressions are pointing to common year. At the moment, we are only able to visualize the events gathered from context and show them in the timeline, but after extending the system, system would able to feed itself to learn and complement its current limitation in recognizing the temporal expressions from tagging level. Also the system will be extensible to have Question Answering functionality by plugin another language pipeline processor and convert it into SPARQL syntax and retrieve the desire results.

V. Conclusion

We have presented AETAS, a novel implemented method that transforms natural language texts to OWL/RDF with resolution of temporal expressions in the context. We have also described a formal semantics for temporal RDF graphs, and a query language for them. Our system allows users to query and browse a RDF database, enriched with user-provided data from unstructured sources.

Since AETAS has been made in SOA approach, it is totally extensible and each component of system can be connected to other external components to further improve the process of semanticizing.

VI. Future work

There are several aspects left for future work. This project in this level can be counted just as a base and core and lot's features can be plugged into the system to enhance its functionality.

6.1 ISLRULE

One of the important future work would be a learning component that uses relationships between linked data to enrich the rule database used by SUTIME Temporal tagger, improving its performance. This feature help the system to understand more about ambiguous tenses like Worker day in the specific day.

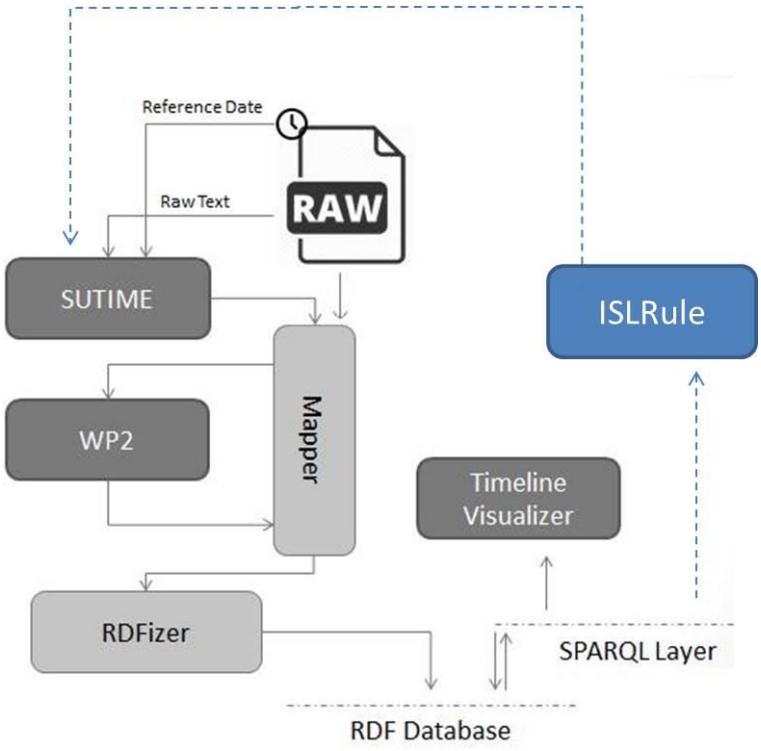

Figure 14: AETAS with ISLRULE Pluggin

As is shown in figure… Intelligence Self-Learning rule (ISLRULE) component has a recursive functionality and connected to RULE base of temporal tagger. So it means whenever temporal tagger found any temporal expressions, if it wouldn't able to normalize them, it call the ISLRULE. Then ISLRULE go back to information database and looking for the expressions before has been stored and if any relationship has been found, it return back to temporal tagger. For example **Epiphany** day (los tres reyes) is the first Sunday after January 1st. so it this fact have been saved in the database, the ISLRULE can realize which year the content is about and calculate the exact day of Epiphany holiday according to the content.

6.2 CRWALER

Another useful component that can be plugged into the system is the web crawler. A System can go through news blogs and website and fetch the content with their data creation time and pass them with their link to AETAS.

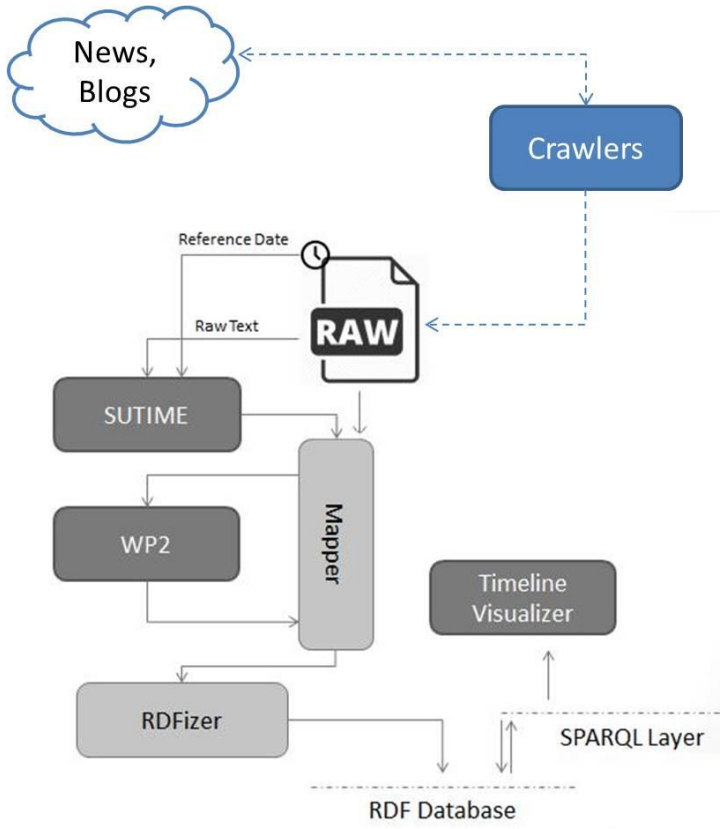

Figure 15: AETAS with Crawler Pluggin

Then AETAS is able to process them and make the current raw text data into linked data and store them for future use.

6.3 PediaSync

PediaSync is component that connect to RDFizer component in one way and has been linked to DBPEDIA from another way. DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web. DBpedia allows you to ask sophisticated queries against Wikipedia, and to link the different data sets on the Web to Wikipedia data.

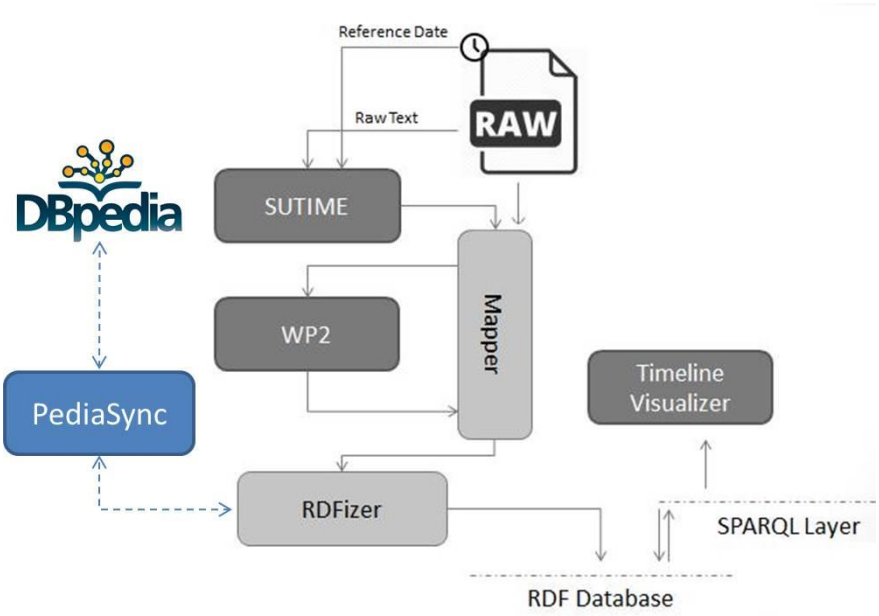

Figure 16:AETAS with PediaSync Pluggin

DBPEDIA has an enrich information about Events, Persons and Spatial data like locations and their information. So AETAS can be more powerful if it has been connected DBPEDIA. Because in the RDFizing process, if any keyword or event or any location appeared, if it would be possible to connected to its relevant description, then the theory of Linked Data has been come into practice. Imagine the sentence "***Obama*** signed the contract Last on this Monday". If AETAS connected to PediaSync, ***Obama*** can be another node that connected to ***Obama*** information on Wikipedia.

Finally, the study of a temporal vocabulary with built-in predicates, such as an order relation, to allow us to specify relationships and restrictions over the time domain.

VII. References

- Omar Alonso, Michael Gertz and Ricardo A. Baeza-Yates and Michael Gertz. 2009. Clustering and Exploring Search Results Using Timeline Constructions. Proceedings of the 18th ACM International Conference on Information and Knowledge Management, pages 97–106.
- Omar Alonso, Jannik Strötgen, Ricardo A. Baeza-Yates and Michael Gertz. 2011. Temporal Information Retrieval: Challenges and Opportunities. Proceedings of TWAW, pages 1-8.
- Isabelle Augenstein, Sebastian Padó, and Sebastian Rudolph. 2012. LODifier: Generating Linked Data from Unstructured Text. In The Semantic Web: Research and Applications. Lecture Notes in Computer Science, 7295: 210-224.
- Sotiris Batsakis and Euripides G. M. Petrakis. 2011. SOWL: A Framework for Handling Spatio-temporal Information in OWL 2.0. Proceeding of RuleML Europe, pages 242-249.
- Matteo Brucato and Danilo Montesi. 2014. Metric Spaces for Temporal Information Retrieval. Proceedings of ECIR, pages 385-397.
- Angel Chang and Christopher D. Manning. 2012. Sutime: a library for recognizing and normalizing time expressions. Proceedings of LREC, pages 3735-3740.
- Miguel Costa, Francisco Couto and Mário Silva. 2014. Learning temporal-dependent ranking models. Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, pages757-766.
- Francesco Draicchio, Aldo Gangemi, Valentina Presutti and Andrea Giovanni Nuzzolese. 2013. FRED: From Natural Language Text to RDF and OWL in One Click. Proceedings of ESWC, pages 263-267.
- Claudio Gutierrez, Carlos A. Hurtado, and Alberto O. Mendelzon, 2003. Formal Aspects of Querying RDF Databases, Proceedings of Workshop Semantic Web and Databases, pages. 293-307.
- Erdal Kuzey and Gerhard Weikum. 2012. Extraction of temporal facts and events from Wikipedia. Proceedings of TempWeb, pages 25-32.
- Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. 2011. Twitinfo: aggregating and visualizing microblogs for event exploration. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 227-236.
- Lluís Padró and Evgeny Stanilovsky. 2012.
- FreeLing 3.0: Towards Wider Multilinguality. Proceedings of LREC, pages 2473-2479.
- James Pustejovsky, José M. Castaño, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, Dragomi and R. Radev. 2003.TimeML: Robust Specification of Event and Temporal Expressions in Text. New Directions in Question Answering, pages 28-34.
- James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003. The TIMEBANK Corpus. Proceedings of Corpus Linguistics Conference, pages 647–656.
- Francisco Nuno Quintiliano Mendonça Carapeto Costa, Processing Temporal Information in Unstructured, DOUTORAMENTO EM INFORMÁTICA ESPECIALIDADE CIÊNCIA DA COMPUTAÇÃO, UNIVERSIDADE DE LISBOA

- Anisa Rula, Matteo Palmonari, Andreas Harth, Steffen Stadtmüller and Andrea Maurino. 2012. On the Diversity and Availability of Temporal Information in Linked Open Data. International Sematic Web Conference, pages 492-507.
- Frank Schilder and Christopher Habel. 2001. From temporal expressions to temporal information: semantic tagging of news messages. Proceedings of the workshop on Temporal and spatial information processing, 13(9):8 pages.
- Jonas Tappolet and Abraham Bernstein. 2009. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. Proceedings of ESWC, pages 308-322