D3.1
Machine Learning and Artificial Intelligence for Digital Interaction Intelligence; State-of-the-art and Guidance Report v2

WP3 Digital Interaction Intelligence Techniques – T3.1 DII software SoTA and guidance

Delivery Date:	*Project Number*:	*Responsible partner*:
M24 - 30/11/2018	ITEA3 Call2 15011	SIVECO ROMANIA SA

DOCUMENT CONTRIBUTORS

Name	Company	Email
George Suciu	BEIA	george@beia.ro
Elena Muelas Cano	HI Iberia Ingeniería y Proyectos	emuelas@hi-iberia.es
Raúl Santos de la Cámara	HI Iberia Ingeniería y Proyectos	rsantos@hi-iberia.es
Yihwa Kim	Taiger	yihwa.kim@taiger.com
Denisa Becheru	SIVECO	Denisa.Becheru@siveco.ro

DOCUMENT HISTORY

Version	Date	Author	Description
0.1	18.04.2018	BEIA	First ToC distribution requesting contributions
0.9	09/07/2018	HIB	Updated ToC, new text for image processing analytics, updated content for Text analysis in section 3.
0.92	15/10/2018	TAIGER	
0.93	23/10/2018	BEIA	Revised version
0.94	17/12/2018	TAIGER	Adding Deep learning used for NLP
1.0	06/01/2019	HIB	Final draft, cleaning up and integrating review process

TABLE OF CONTENTS

1. Introduction

The execution of a research project such as SoMeDi requires its participants to remain at the cutting edge of technology not only during the preparation or at the start of the project, but at all times during its execution. Thus, we propose task T3.3 whose overall objective is to provide a living or dynamic document that describes the state of the art in the different sub domains of knowledge that comprise Digital Interaction intelligence and a guidance report captured from SoMeDi experience all throughout the project. The output of Task 3.1 consists of this current deliverable D3.1.

Deliverable D3.1 has had two iterations with the objective to have an ongoing view of state-of-the-art in DII domain. The first iteration compiled the current trends in algorithms and open source tools from machine learning, pattern recognition, natural language processing and opinion mining. This second iteration, delivered one year later, updates the state-of-the-art section and will provide a guidance report, construct based on our experience gathered during the first two years of SoMeDi project.

The current document represents the final iteration of D3.1. The material presented here is the culmination of the work performed by the responsible partners: they conducted a literature review and a field research, based on their knowledge and experience, to identify the most suitable algorithms and the available tools that can be used to develop the SoMeDi technology toolkit. The field research was primarily conducted in the domains that were complementary to the ones in D3.1 v1: data extraction, NLP technologies and metadata extraction from images.

This final iteration of D3.1 document is organised as follows:

- Section 2 describes algorithms and open tools available for Data Extraction.
- Section 3 describes algorithms and open tools available for Natural Language Processing. This analysis complements the first iteration results in D3.1 v1.
- Section 4 describes algorithms and open tools available to perform metadata and data extraction from images captured in the social media. This is entirely new not discussed in v1 of D3.1.
- Section 5 concludes the document and presents brief hints of the future of the work in SoMeDi related to these fields.

2. Data Extraction

Data extraction is the act or process of retrieving data out of (usually unstructured or poorly structured) data sources for further data processing or data storage (data migration). The import into the intermediate extracting system is thus usually followed by data transformation and possibly the addition of metadata prior to exporting to another stage in the data workflow.

2.1. Data Extraction Techniques

In order to acquire content from the internet there are basically four techniques:

Crawlers

The key points of crawlers are scalability and volume. They follow links from web pages around the Internet (or within a website) and download pages. They can be distributed across many machines to download tens of thousands of web pages.

Heritrix[1] – from the Open Internet Archive, Nutch[2] – from Apache, and Aspider[3] – from Search Technologies, are several popular solutions when it comes to crawling through internet content.

Scrapers

Scrapers center on extracting content. Scrapers are typically less scalable and more hand-tuned than crawlers and focus instead on extracting content (such as numeric and metadata information) from the web pages they download. When is required to obtain structured data from web pages based on presentation structure, a scraper may be the best choice.

Some common scrapers include: Scrapy[4] - a Python-based scraper which has a hosted cloud-based version and a graphical tool to help create scrapers; Octoparse[5] – an MS-Windows scraper with visual tools to implement scraping; Apifier[6] – a cloud-based JavaScript scraper; Content Grabber[7] – a screen scraper with scripting, dynamic parameters, and ability to handle SSO (Single Sign-On) cookies and proxies; and UiPath[8] – which is more of a larger "automation framework" offering a screen scraping component.

As previously specified, scrapers are able to extract structured content where it is structured on the web page, in other words, based on HTML tagging, JSON structures, etc. But they require more work and programming than a crawler (which is simply "point and go"), and the output is more structured and immediately useful.

Browser Automation

Browser automation retrieves and renders the page like a web browser. Browser automation tools actually run the JavaScript pulled from the web pages and render the HTML (and other

[1] https://webarchive.jira.com/wiki/spaces/Heritrix
[2] http://nutch.apache.org/
[3] https://www.searchtechnologies.com/blog/aspider-web-crawler-intranet-search
[4] https://scrapy.org/
[5] https://www.octoparse.com/
[6] https://www.apify.com/
[7] https://contentgrabber.com/
[8] https://www.uipath.com/automate/screen-scraping

data structures). They can then be combined with custom scripting to explore the results and download content which might otherwise be inaccessible.

Some common browser automation tools include: Splash[9], PhantomJS[10], Selenium[11], Nightmare[12].

Third-Party APIs

Third-party APIs are required for third-party content providers. In order to access data from third-party content providers, such as Thomson Reuters, LexisNexis, Bing, Factiva, NewsCred, etc., it's necessary to use the APIs they provide.

Fortunately, these providers have taken effort to deliver good structured data, and so using these APIs will typically require a lot less time than using a scraper or browser automation tool.

2.2. Data Models

A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized and manipulated. The most popular example of a database model is the relational model, which uses a table-based format.

Data Warehouse

A data warehousing is a technique for collecting and managing data from varied sources to provide meaningful business insights. It is a blend of technologies and components which allows the strategic use of data.

A Data Warehouse works as a central repository where information arrives from one or more data sources. Data flows into a data warehouse from the transactional system and other relational databases.

Data may be:

- Structured
- Semi-structured
- Unstructured data

The data is processed, transformed, and ingested so that users can access the processed data in the Data Warehouse through Business Intelligence tools, SQL clients, and spreadsheets. A data warehouse merges information coming from different sources into one comprehensive database.

Types of Data Warehouse

Three main types of Data Warehouses are:

[9] http://splash.readthedocs.io/en/stable/index.html
[10] http://phantomjs.org/
[11] https://www.seleniumhq.org/
[12] https://github.com/segmentio/nightmare

1. Enterprise Data Warehouse:

Enterprise Data Warehouse is a centralized warehouse. It provides decision support service across the enterprise. It offers a unified approach for organizing and representing data. It also provide the ability to classify data according to the subject and give access according to those divisions.

2. Operational Data Store:

Operational Data Store, which is also called ODS, are nothing but data store required when neither Data warehouse nor OLTP systems support organizations reporting needs. In ODS, Data warehouse is refreshed in real time. Hence, it is widely preferred for routine activities like storing records of the Employees.

3. Data Mart:

A data mart is a subset of the data warehouse. It specially designed for a particular line of business, such as sales, finance, sales or finance. In an independent data mart, data can collect directly from sources.

Data Warehouse Tools

There are many Data Warehousing tools are available in the market. Here, are some most prominent one:

1. MarkLogic:

MarkLogic[13] is useful data warehousing solution that makes data integration easier and faster using an array of enterprise features. This tool helps to perform very complex search operations. It can query different types of data like documents, relationships, and metadata.

2. Oracle:

Oracle[14] is the industry-leading database. It offers a wide range of choice of data warehouse solutions for both on-premises and in the cloud. It helps to optimize customer experiences by increasing operational efficiency.

3. Amazon RedShift:

Amazon Redshift[15] is Data warehouse tool. It is a simple and cost-effective tool to analyze all types of data using standard SQL and existing BI tools. It also allows running complex queries against petabytes of structured data, using the technique of query optimization.

4. MS SSIS

[13] http://developer.marklogic.com/products

[14] https://www.oracle.com/index.html

[15] https://aws.amazon.com/redshift/?nc2=h_m1

SQL Server Integration Services[16] is a Data warehousing tool that used to perform ETL operations; i.e. extract, transform and load data. SQL Server Integration also includes a rich set of built-in tasks.

Features:

- Tightly integrated with Microsoft Visual Studio and SQL Server
- Easier to maintain and package configuration
- Allows removing network as a bottleneck for insertion of data
- Data can be loaded in parallel and various locations
- It can handle data from different data sources in the same package
- SSIS consumes data which are difficult like FTP, HTTP, MSMQ, and Analysis services, etc.
- Data can be loaded in parallel to many varied destinations

Multidimensional Data Models

Multidimensional data model stores data in the form of data cube. Mostly, data warehousing supports two or three-dimensional cubes. A data cube allows data to be viewed in multiple dimensions. A dimensions are entities with respect to which an organization wants to keep records. For example in store sales record, dimensions allow the store to keep track of things like monthly sales of items and the branches and locations. A multidimensional databases helps to provide data-related answers to complex business queries quickly and accurately.

Data warehouses and Online Analytical Processing (OLAP) tools are based on a multidimensional data model. OLAP in data warehousing enables users to view data from different angles and dimensions.

Schemas for Multidimensional Data Model are:

Star Schemas for Multidimensional Model

The simplest data warehouse schema is star schema because its structure resembles a star. Star schema consists of data in the form of facts and dimensions. The fact table present in the center of star and points of the star are the dimension tables. In star schema fact table contain a large amount of data, with no redundancy. Each dimension table is joined with the fact table using a primary or foreign key.

[16] https://www.microsoft.com/en-us/download/details.aspx?id=39931

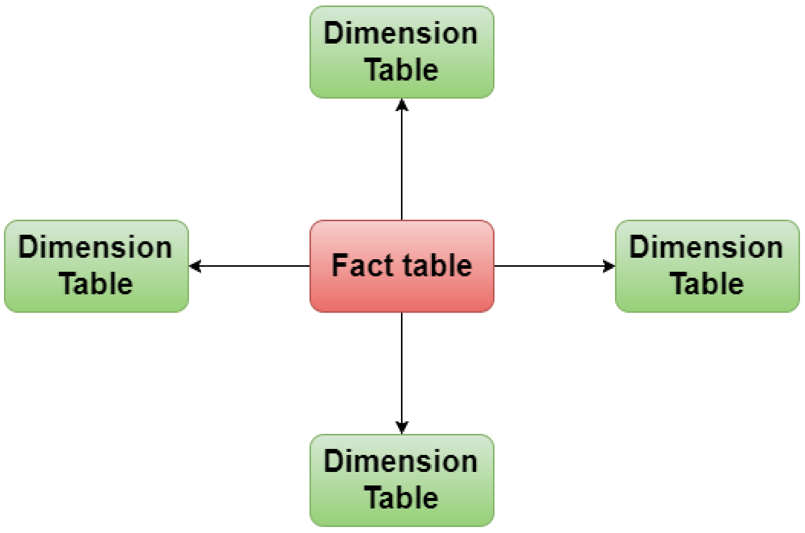


FIGURE 1. – STAR SCHEMA FOR MULTIDIMENSIONAL MODEL

Snowflake Schemas for Multidimensional Model

The snowflake schema is a more complex than star schema because dimension tables of the snowflake are normalized. The snowflake schema is represented by centralized fact table which is connected to multiple dimension table and this dimension table can be normalized into additional dimension tables.

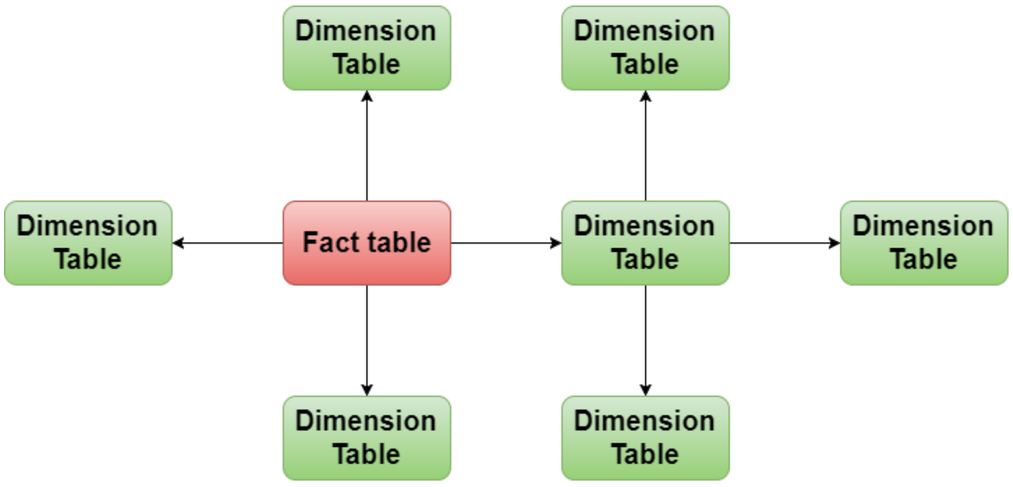


FIGURE 2. –SNOWFLAKE SCHEMA FOR MULTIDIMENSIONAL MODEL

Fact constellation Schemas for Multidimensional Modal

A fact constellation can have multiple fact tables that share many dimension tables. This type of schema can be viewed as a collection of stars, Snowflake and hence is called a galaxy schema or a fact constellation.

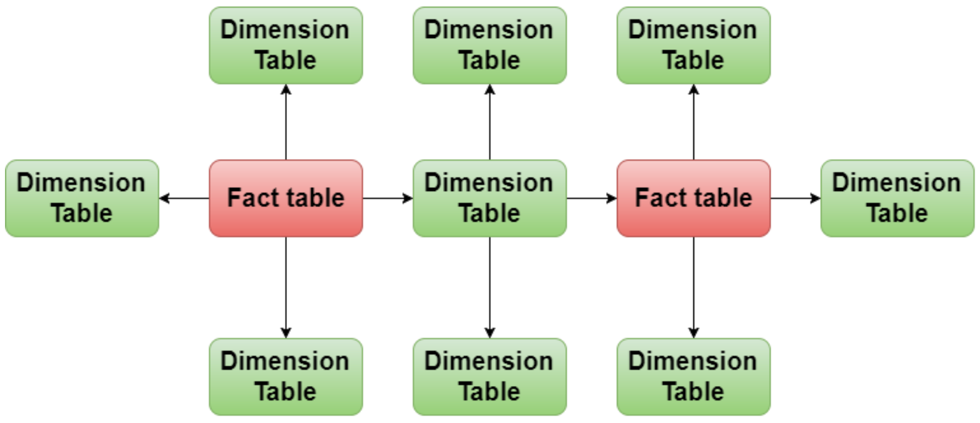


FIGURE 3. – FACT CONSTELLATION SCHEMA FOR MULTIDIMENSIONAL MODEL

3. NLP Techniques

In this section, we describe a structural view of the concepts (see Figure 1) involved in Natural Language Processing (NLP) solutions, whether it being text extraction or sentiment analysis.

NLP is quickly becoming an essential skill for modern-day organizations to gain a competitive edge. It has become the essential tool for many new business functions, from chatbots and question-answering systems to sentiment analysis, compliance monitoring, and BI (Business Intelligence) and analytics of unstructured and semi-structured content.

Consider all the unstructured content that can bring significant insights – queries, email communications, social media, videos, customer reviews, customer support requests, etc. Natural Language Processing (NLP) tools and techniques help process, analyse, and understand unstructured "big data" in order to operate effectively and proactively.

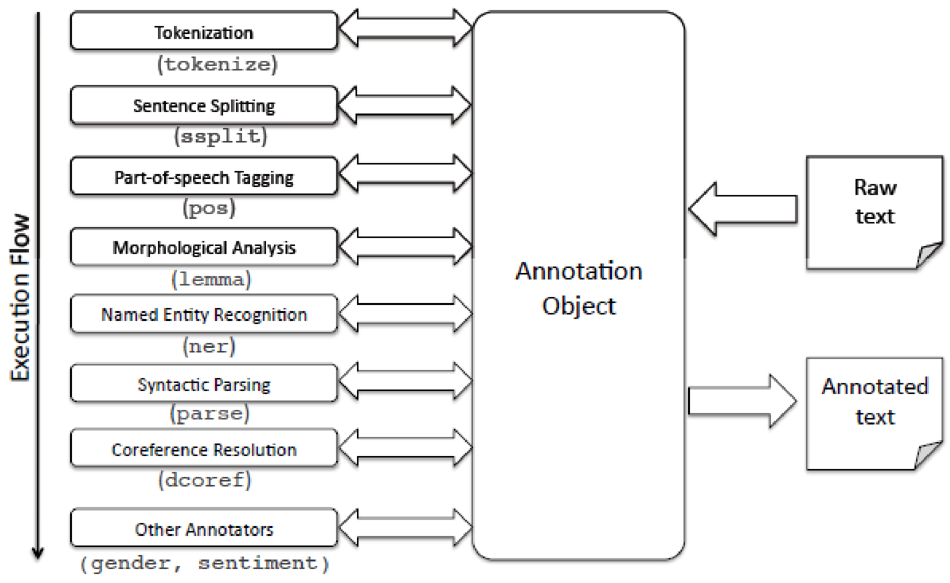


FIGURE 4. - NATURAL LANGUAGE PROCESSING PROCESSES WORKFLOW[17]

In many use cases, the content with the most important information is written down in a natural language (such as English, German, Spanish, Chinese, etc.) and not conveniently tagged. To extract information from this content you will need to rely on some levels of text mining, text extraction, or possibly full-up natural language processing (NLP) techniques.

Typical full-text extraction for Internet content includes:

- Extracting entities – such as companies, people, dollar amounts, key initiatives, etc.,
- Categorizing content – positive or negative (e.g. sentiment analysis), by function, intention or purpose, or by industry or other categories for analytics and trends,
- Clustering content – to identify main topics of discourse and/or to discover new topics,

[17] Manning et. al, The Stanford CoreNLP Natural Language Processing Toolkit, Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp 55-60

- Fact extraction – to fill databases with structured information for analysis, visualization, trending, or alerts,
- Relationship extraction – to fill out graph databases to explore real-world relationships.

3.1. Explaining NLP concepts

Metadata mining: Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. Data mining is the process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.

The following NLP concepts (phases) specific to text extraction and sentiment analysis solutions, described in section 3.2, combine data extraction and annotation techniques. We will present in the following table an overview of several solutions characteristics to each of the phases shown in Figure 4.

Concepts	Description	Solutions
Structure extraction	Identifying fields and blocks of content based on tagging.	
Syntactic markers	Identify and mark sentence, phrase, and paragraph boundaries; are important when doing entity extraction and NLP since they serve as useful breaks within which analysis occurs.	- Open NLP Apache sentence and paragraph[18] - Lucene Segmenting Tokenizer[19]
Language identification	Will detect the human language for the entire document and for each paragraph or sentence. Language detectors are critical to determining what linguistic algorithms and dictionaries to apply to the text.	- Google Language Detector[20] - Optimize Language Detector[21] - Chromium Compact Language Detector[22] - API methods: Bing Language Detection API, IBM Watson Language Identification[23], Google Translation API for Language Detection[24], Apertium API[25]

[18] https://opennlp.apache.org/docs/
[19] https://lucene.apache.org/core/6_2_1/analyzers-common/org/apache/lucene/analysis/util/SegmentingTokenizerBase.html
[20] https://github.com/shuyo/language-detection/blob/wiki/ProjectHome.md
[21] https://github.com/optimaize/language-detector
[22] https://github.com/mikemccand/chromium-compact-language-detector
[23] https://www.ibm.com/watson/developercloud/language-translator/api/v2/curl.html?curl
[24] https://cloud.google.com/translate/docs/detecting-language
[25] http://wiki.apertium.org/wiki/Apertium-apy

Tokenization	To divide up character streams into tokens which can be used for further processing and understanding. Tokens can be words, numbers, identifiers or punctuation (depending on the use case).	- Lucene Analyzers[26] - Open NLP Tokenizer[27]
Acronym normalization and tagging	Acronyms can be specified as “I.B.M.” or “IBM” so these should be tagged and normalized.	- Search Technologies[28]
Lemmatization / Stemming	Lemmatization reduces word variations to simpler forms, also uses a language dictionary to perform an accurate reduction to root words. Stemming is the process of converting the words of a sentence to its non-changing portions. Whereas lemmatization is the process of converting the words of a sentence to its dictionary form. Lemmatization is strongly preferred to stemming if available.	- Basis Technologies[29] - Open Source Lucene analyzers[30]
Decompounding	For some languages (typically Germanic, Scandinavian, and Cyrillic languages), compound words will need to be split into smaller parts to allow for accurate NLP.	- Basis Technologies
Entity extraction	Identifying and extracting entities (people, places, companies, etc.) is a necessary step to simplify downstream processing.	- Regex extraction – good for phone numbers, ID numbers, e-mail addresses, URLs, etc. - Dictionary extraction – recommended for known entities, such as colors, units, sizes, employees, business groups, drug names, products, brands.

[26] http://lucene.apache.org/core/6_5_0/analyzers-common/index.html
[27] https://opennlp.apache.org/docs/1.8.2/apidocs/opennlp-tools/opennlp/tools/tokenize/Tokenizer.html
[28] https://www.searchtechnologies.com/
[29] https://www.basistech.com/
[30] http://lucene.apache.org/core/6_5_0/analyzers-common/index.html

		- Complex pattern-based extraction – recommended for people's names (made of known components), business names (made of known components) and context-based extraction scenarios (e.g., extract an item based on its context) - Statistical extraction – uses statistical analysis to do context extraction.
Phrase extraction	Extracts sequences of tokens (phrases) that have a strong meaning which is independent of the words when treated separately. These sequences should be treated as a single unit when doing NLP.	- *Part of speech tagging* – identifies phrases from noun or verb clauses - *Statistical phrase extraction* - identifies token sequences which occur more frequently than expected by chance - - *Hybrid* - uses both techniques together and tends to be the most accurate method.

TABLE 1.- DESCRIPTION OF NLP CONCEPTS

The level of content understanding is categorized as:

- **Macro Understanding** – provides a general understanding of the document as a whole. Typically performed with statistical techniques, it is used for: clustering, categorization, similarity, topic analysis, word clouds, and summarization;
- **Micro Understanding** – extracts information gained from individual phrases or sentences. It is used for: extracting facts, entities (see above), entity relationships, actions, and metadata fields.

Having in mind the processes involved in NLP projects, we apprehend the value of metadata and how it is distinguished in SoMeDi platform for NLP backend services.

Basically, it all comes to Semantic Annotation of Documents. Information Extraction (IE) for the Semantic Web is structured as follows:

- Traditional IE is based on a flat structure, e.g. recognising Persons, Locations, Organisations, Date, Time etc.
- For the Semantic Web, we need information which is organized in a hierarchical structure. The idea is that we will attach semantic metadata to the documents, pointing to concepts in an ontology.
- Information can be exported as an ontology annotated with instances, or as text annotated with links to the ontology, in Figure 2 displayed below, we have a graphic example of semantic annotation.

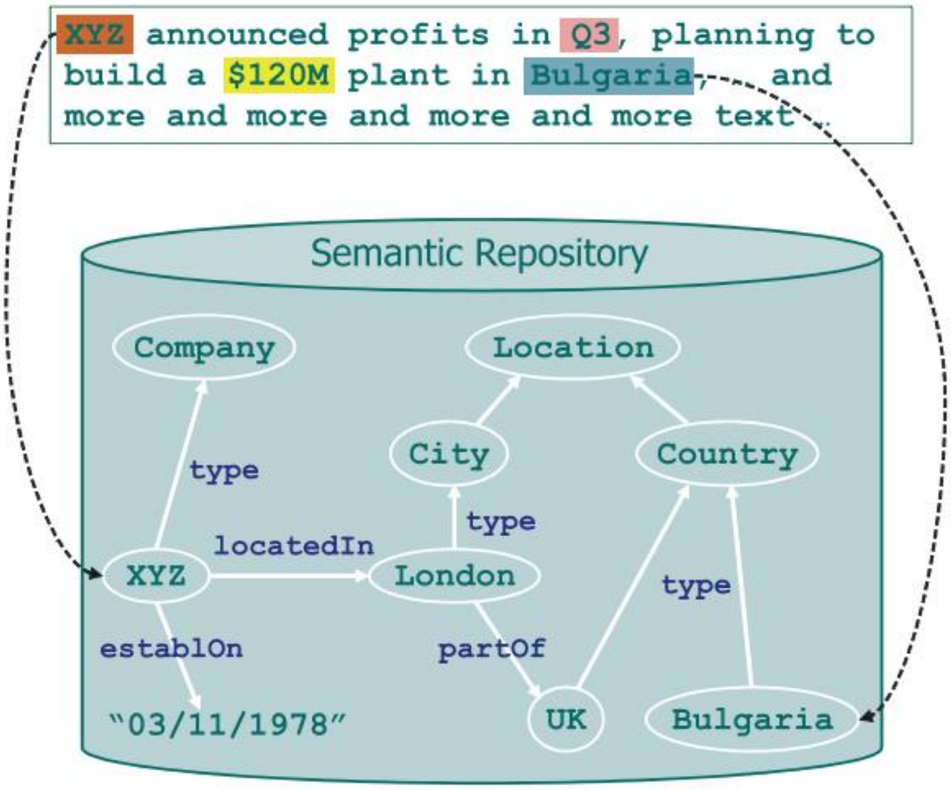
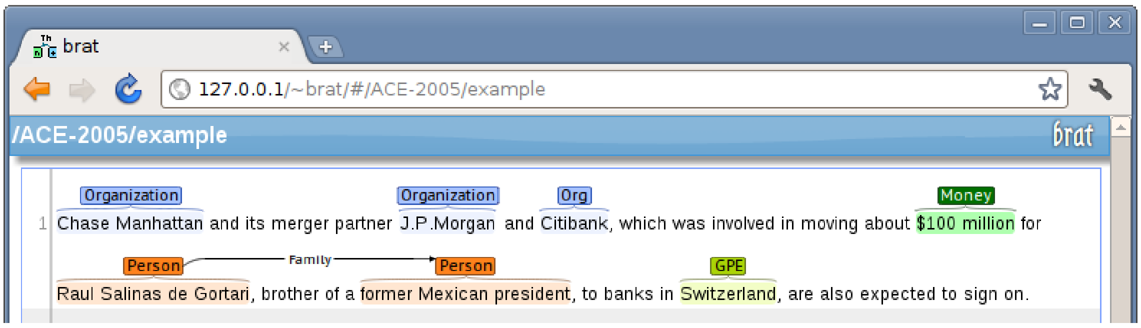


FIGURE 5. – SEMANTIC ANNOTATION

Brat[31] is a web-based tool for text annotation; that is for adding notes to existing text documents. Brat is designed in particular for structured annotation, where the notes are not freeform text but have a fixed form that can be automatically processed and interpreted by a computer.

The following diagram, Figure 3, shows a simple example where a sentence has been annotated to identify mentions of some real-world entities (things) and their types, and a relation between two.



Chase Manhattan and its merger partner J.P.Morgan and Citibank, which was involved in moving about $100 million for Raul Salinas de Gortari, brother of a former Mexican president, to banks in Switzerland, are also expected to sign on.

FIGURE 6. –SEMANTIC ANNOTATION WITH BRAT

MMAX2[32] is an XML-based tool that is particularly useful for anaphora annotation.

31 http://brat.nlplab.org/introduction.html

32 http://mmax2.net

The CLaRK[33] system is a fairly robust system for encoding syntactic annotation.

Other annotation tools widely used are: APACHE UIMA [34] (Unstructured Information Management Architecture), WebAnno[35] - a flexible, web-based and visually supported system for distributed annotations, and Callisto[36].

In the next section, we will describe several development and services frameworks for Natural Language Processing, but first, we will perform a comparison analysis regarding the levels of content understanding (see Table 2).

Macro Understanding	Micro Understanding
• Classifying / categorizing / organizing records • Clustering records • Extracting topics • General sentiment analysis • Record similarity, including finding similarities between different types of records (for example, job descriptions to resumes / CVs) • Keyword / keyphrase extraction • Duplicate and near-duplicate record detection • Summarization / key sentence extraction • Semantic search	• Extracting acronyms and their definitions • Extracting citation references to other documents • Extracting key entities (people, company, product, dollar amounts, locations, dates). Note that extracting “key” entities is not the same as extracting “all” entities (there is some discrimination implied in selecting what entity is ‘key’) • Extracting facts and metadata from full text when it’s not separately tagged in the web page • Extracting entities with sentiment (e.g. positive sentiment towards a product or company) • Identifying relationships such as business relationships, target / action / perpetrator, etc. • Identifying compliance violations, statements which show possible violation of rules • Extracting statements with attribution, for example, quotes from people (who said what) • Extracting rules or requirements, such as contract terms, regulation requirements, etc.

TABLE 2.- MACRO / MICRO UNDERSTANDING FEATURES

There are three approaches to performing extraction that provides micro understanding:

1. **Top Down** - determine Part-of-Speech, then understand and diagram the sentence into clauses, nouns, verbs, object and subject, modifying adjectives and adverbs, etc., then traverse this structure to identify structures of interest;

33 http://www.bultreebank.org/clark/index.html
34 https://uima.apache.org/
35 https://webanno.github.io/webanno/
36 http://mitre.github.io/callisto/

2. **Bottom Up** – create lots of patterns, match the patterns to the text and extract the necessary facts. Patterns may be manually entered or may be computed using text mining;
3. **Statistical** – similar to bottom-up, but matches patterns against a statistically weighted database of patterns generated from tagged training data.

In Table 3 below, we performed a SWOT analysis of the three above mentioned methods.

Micro understanding Method	Advantages	Disadvantages
Top Down	• can handle complex, never-seen-before structures and patterns	• hard to construct rules, brittle, often fails with variant input, may still require substantial pattern matching even after parsing.
Bottom Up	• easy to create patterns, can be done by business users, does not require programming, easy to debug and fix, runs fast, matches directly to desired outputs	• requires on-going pattern maintenance, cannot match on newly invented constructs
Statistical	• patterns are created automatically, built-in statistical trade-offs	• requires generating extensive training data (1000's of examples), will need to be periodically retrained for best accuracy, cannot match on newly invented constructs, harder to debug.

TABLE 3.- MICRO UNDERSTANDING METHODS SWOT ANALYSIS

3.2. Development and service frameworks for NLP

3.2.1. Development frameworks for NLP

The use of open source NLP development frameworks is favoured due to their full of high-quality libraries to solve common problems in text processing like sentiment analysis, topic identification, automatic labeling of content, and more. More importantly, open source also provides many building block libraries that make it easy for you to innovate without having to reinvent the wheel.

Also, other advantages when working with open source frameworks include:

a) access to the source code that means – having the possibility to understand the algorithms used, and - having access to change the code according to the project requirements;

b) a reduced cost for development.

Stanford's Core NLP Suite[37] consists of a GPL-licensed framework of tools for processing English, Chinese, Arabic, French, German and Spanish. It includes tools for tokenization

[37] https://stanfordnlp.github.io/CoreNLP/

(splitting of text into words), part of speech tagging, grammar parsing (syntactic analysis), named entity recognition, and more.

Natural Language Toolkit[38] is a Python programming toolkit. Similar to the Stanford tools, it includes capabilities for tokenizing, parsing, and identifying named entities as well as many more features.

Apache Lucene[39] and Solr are not quite technically targeted at solving NLP problems, Lucene and Solr contain a powerful number of tools for working with text, ranging from advanced string manipulation utilities to powerful and flexible tokenization libraries to blazing fast libraries for working with finite state automata.

Apache OpenNLP[40] uses a different underlying approach from Stanford's. The OpenNLP project is an Apache-licensed suite of tools to do tasks like tokenization, part of speech tagging, parsing, and named entity recognition. While not necessarily state of the art anymore in its approach, it remains a solid choice, that is easy to get up and running.

GATE[41] and Apache UIMA are suitable for building complex NLP workflows which need to integrate several different processing steps. In these cases, it is recommended to work with a framework like GATE or UIMA that standardizes and abstracts much of the repetitive work that goes into building a complex NLP application. GATE relies on a configurable bottom-up approach; and it is much easier to work with, However configurations must still be created by programmers (not by business users).

3.2.2. Service frameworks for NLP

The three leading cloud computing vendors, AWS, Microsoft Azure and Google Cloud, are also the major NLP service frameworks providers. Developing an AI Framework for Natural Language Processing is available through MLaaS - Machine Learning as a Service, automated and semi-automated cloud platforms that cover most infrastructure issues such as data pre-processing, model training, and model evaluation, with further prediction. Prediction results can be bridged with the internal IT infrastructure through REST APIs.

Amazon NLP services framework

Amazon Comprehend[42] is another NLP set of APIs that aim at different text analysis tasks. Currently, Comprehend supports:

- Entities extraction (recognizing names, dates, organizations, etc.);
- Key phrase detection;
- Language recognition;
- Sentiment analysis (how positive, neutral, or negative a text is);
- Topic modeling (defining dominant topics by analyzing keywords).

This service will help in projects like analyzing social media responses, comments, and other big textual data that's not amenable to manual analysis.

Amazon Translate[43], as the name states, is a Translate service that translates texts. It uses neural networks which provides better translation quality than the rule-based translation approaches, according to Amazon. Unfortunately, the current version supports translation

[38] http://www.nltk.org/
[39] http://lucene.apache.org/
[40] http://opennlp.apache.org/
[41] https://gate.ac.uk/
[42] https://aws.amazon.com/comprehend/
[43] https://aws.amazon.com/translate/?nc2=h_m1

from only six languages into English and from English into those six. The languages are Arabic, Chinese, French, German, Portuguese, and Spanish.

MICROSOFT AZURE COGNITIVE SERVICES

Just like Amazon, Microsoft suggests high-level APIs, Cognitive Services[44], that can be integrated with a private IT infrastructure and perform tasks with no data science expertise needed.

The language group of APIs focuses on textual analysis similar to Amazon Comprehend:

- Language Understanding Intelligent Service is an API that analyzes intentions in text to be recognized as commands (e.g. "run YouTube app" or "turn on the living room lights");
- Text Analysis API for sentiment analysis and defining topics;
- Bing Spell Check;
- Translator Text API;
- Web Language Model API that estimates probabilities of words combinations and supports word autocompletion;
- Linguistic Analysis API used for sentence separation, tagging the parts of speech, and dividing texts into labeled phrases.

GOOGLE CLOUD SERVICES

Google's set of APIs mainly mirrors with what Amazon and Microsoft Azure suggest, it has some interesting and unique things to look at.

Cloud natural language API[45] is almost identical in its core features to Comprehend by Amazon and Language by Microsoft.

- Defining entities in text;
- Recognizing sentiment;
- Analyzing syntax structures;
- Categorizing topics (e.g. food, news, electronics, etc.).

Cloud translation API[46] can be used to employ Google Translate and it includes over a hundred languages and also has automatic language detection feature.

Table 4 below displays the text processing APIs comparison.

Text Analytics	Amazon	Microsoft	Google
Entities Extraction	✓	✓	✓
Key Phrase Extraction	✓	✓	✓

[44] https://azure.microsoft.com/en-us/services/cognitive-services/
[45] https://cloud.google.com/natural-language/
[46] https://cloud.google.com/translate/

Language Recognition	>100 languages	120 languages	>100 languages
Topics Extraction	✓	✓	✓
Spell Check	✗	✓	✗
Autocompletion	✗	✓	✗
Intentions Analysis	✓	✓	✓
Sentiment Analysis	✓	✓	✓
Syntax Analysis	✗	✓	✓
Tagging Parts of Speech	✗	✓	✓
Filtering inappropriate Content	✗	✓	✓
Translation	6 languages	>60 languages	>100 languages
Chatbot Toolset	✓	✓	✓

TABLE 4.- TEXT PROCESSING APIs COMPARISON [47]

3.3.Metadata understanding in NLP service frameworks

After analyzing the available solutions and concepts involved in Natural Language Processing, we will relate to the two experimented solutions which are planned to be integrated into Somedi platform for text extraction, text translation, and sentiment analysis tasks, and present how metadata links to each of the steps required for these tasks.

Google Cloud Natural Language

As mentioned above, the Google Cloud Natural Language API supports a variety of languages. These languages are specified within a request using the optional language parameter. Language code parameters conform to ISO-639-1[48] or BCP-47[49] identifiers. If you do not specify a language parameter, then the language for the request is auto-detected by the Natural Language API.

The Natural Language API has several methods for performing analysis and adding annotations on a given text. Each level of analysis provides valuable information for language understanding. These methods are listed below:

- **Sentiment analysis** inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine the writer's attitude as positive, negative, or neutral. Sentiment analysis is performed through the analyzeSentiment method;
- **Entity analysis** inspects the given text for known entities (Proper nouns such as public figures, landmarks, and so on. Common nouns such as restaurant, stadium, and so on.)

[47] https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai/
[48] https://en.wikipedia.org/wiki/ISO_639-1
[49] https://tools.ietf.org/html/bcp47

and returns information about those entities. Entity analysis is performed with the analyzeEntities method;

- **Entity sentiment analysis** inspects the given text for known entities (proper nouns and common nouns), returns information about those entities, and identifies the prevailing emotional opinion of the entity within the text, especially to determine a writer's attitude toward the entity as positive, negative, or neutral. Entity analysis is performed with the analyzeEntitySentiment method;
- **Syntactic analysis** extracts linguistic information, breaking up the given text into a series of sentences and tokens (generally, word boundaries), providing further analysis on those tokens. Syntactic Analysis is performed with the analyzeSyntax method;
- **Content classification** analyzes text content and returns a content category for the content. Content classification is performed by using the classifyText method.

Each API call also detects and returns the language, if a language is not specified by the caller in the initial request. Additionally, if required to perform several natural language operations on a given text using only one API call, the annotateText request can also be used to perform sentiment analysis and entity analysis.

The Natural Language API is a REST API, and consists of JSON requests and responses. A simple Natural Language JSON Entity Analysis request appears below:

```
{

"document":{
    "type":"PLAIN_TEXT",
    "language": "EN",
    "content":"'Lawrence of Arabia' is a highly rated film
biography about \
                British Lieutenant T. E. Lawrence. Peter
O'Toole plays \
                Lawrence in the film."
  },
  "encodingType":"UTF8"

}
```

These fields are explained below:

- document contains the data for this request, which consists of the following sub-fields:
- type - document type (HTML or PLAIN_TEXT)
- language - (optional) the language of the text within the request. If not specified, language will be automatically detected. For information on which languages are supported by the Natural Language API, see Language Support. Unsupported languages will return an error in the JSON response.
- Either content or gcsContentUri which contain the text to evaluate. If passing content, this text is included directly in the JSON request (as shown above). If passing gcsContentUri, the field must contain a URI pointing to text content within Google Cloud Storage.
- encodingType - (required) the encoding scheme in which returned character offsets into the text should be calculated, which must match the encoding of the passed text. If this parameter is not set, the request will not result as error, but all such offsets will be set to -1.

MICROSOFT AZURE TEXT ANALYTICS API

The Text Analytics API is a suite of text analytics web services built with machine learning algorithms. The API can be used to analyze unstructured text for tasks such as sentiment analysis, key phrase extraction and language detection. No training data is needed to use this API; just feed the required text data. This API uses advanced natural language processing techniques to deliver best in class predictions.

Calls to the three Text Analytics API are HTTP POST/GET calls, which can be formulated in any language. In this case, we used REST and Postman to demonstrate key concepts. Each request must include an access key and an HTTP endpoint. The endpoint specifies the region chosen during sign up, the service URL, and a resource used on the request: sentiment, keyphrases, languages.

It is important to specify that Text Analytics is stateless so there are no data assets to manage. The text sample is uploaded, analyzed upon receipt, and results are returned immediately to the calling application.

The structure of the request URL is as follows:

```
https://[location].api.cognitive.microsoft.com/text/analytics/v2.0/keyPhrases
```

Input must be JSON in raw unstructured text. XML is not supported. The schema is simple, consisting of the elements described in the following table (Table 5). It is possible to submit the same documents for all three operations: sentiment, key phrase, and language detection. (The schema is likely to vary for each analysis in the future.)

Element	Valid values	Required (or Optional)	Usage
id	The data type is string, but in practice, document IDs tend to be integers.	Required	The system uses the IDs provided to structure the output. Language codes, key phrases, and sentiment scores are generated for each ID in the request.
text	Unstructured raw text, up to 5,000 characters.	Required	For language detection, text can be expressed in any language. For sentiment analysis and key phrase extraction, the text must be in a supported language.

Element	Valid values	Required (or Optional)	Usage
language	2-character ISO 639-1 code for a supported language	Varies	Required for sentiment analysis and key phrase extraction, optional for language detection. There is no error if you exclude it, but the analysis is weakened without it. The language code should correspond to the provided text.

TABLE 5.- JSON SCHEMA DEFINITION

The Language Detection API evaluates text input and for each document returns language identifiers with a score indicating the strength of the analysis. Text Analytics recognizes up to 120 languages.

This capability is useful for content stores that collect arbitrary text, where language is unknown. It's possible to parse the results of this analysis to determine which language is used in the input document. The response also returns a score which reflects the confidence of the model (a value between 0 and 1).

JSON documents in the following format are required: id, text. Text must be under 5,000 characters per document, and 1,000 items (Documents or IDs) are supported per collection. The collection is submitted in the body of the request. Following is an example of a response after the text analysis and language detection steps.

```
{
  "languageDetection": {
    "documents": [
      {
        "id": "bebe543b-cfb0-4fde-93e4-d565d2cec547",
        "detectedLanguages": [
          {
            "name": "English",
            "iso6391Name": "en",
            "score": 1.0
          }
        ]
      }
    ],
    "errors": []
  },
  "keyPhrases": {
    "documents": [
      {
        "id": "bebe543b-cfb0-4fde-93e4-d565d2cec547",
        "keyPhrases": [
          "writing novels",
          "romance"
        ]
      }
    ],
    "errors": []
  },
  "sentiment": {
    "documents": [
      {
        "id": "bebe543b-cfb0-4fde-93e4-d565d2cec547",
        "score": 0.14186686277389526
      }
    ],
    "errors": []
  }
}
```

3.4 Deep learning for Natural Language Processing

3.4.1 What is deep learning

Deep learning is an approach to AI. It is a type of machine learning, a technique that allows computer systems to improve with experience and data. Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstracted representations computed in terms of less abstract ones (Goodfellow, et al., 2016).

3.4.2 Deep learning and NLP

Natural language processing (NLP) is the use of human languages, by a computer. Computer programs typically read and emit specialized languages designed to allow efficient and and unambiguous parsing by simple programs. More naturally occurring languages are often ambiguous and defy formal description. Natural language processing includes applications such as machine translation, in which the learner must read a sentence in one human language and emit an equivalent sentence in another human language. Many NLP applications are based on language models that define a probability distribution over sequences of words, characters or bytes in a natural language (Goodfellow, et al., 2016).

Deep learning architectures and algorithms have already made impressive advances in fields such as computer vision and pattern recognition. Following this trend, recent NLP research is now increasingly focusing on the use of new deep learning methods. For decades, machine learning approaches targeting NLP problems have been based on shallow models (e.g., SVM and logistic regression) trained on very high dimensional and sparse features. In the last few years, neural networks based on dense vector representations have been producing superior results on various NLP tasks. This trend is sparked by the success of word embeddings (Mikolov, et al., 2010) (Mikolov, et al., 2013) and deep learning methods (Socher, et al., 2013). Deep learning enables multi-level automatic feature representation learning. In contrast, traditional machine learning based NLP systems liaise heavily on hand-crafted features. Such hand-crafted features are time-consuming and often incomplete (Young, et al., 2018).

3.4.3 Embeddings

In order for words to be represented as input to the deep learning task, it needs to be converted into a vector based representation. Embeddings are dense representation of words, where each word is represented as dense vector, whereas a one-hot encoding would be a sparse representation of words.

In word embeddings, raw symbols are viewed as points in a space of dimension equal to the vocabulary size. The word representations embed those points in a feature space of lower dimension. In the original space, every word is represented by a one-hot vector, so every pair of words is at Euclidean distance $\sqrt{2}$ from each other. In the embedding space, words that frequently appear in similar contexts (or any pair of words sharing some 'features' learned by the mode) are close to each other (Goodfellow, et al., 2016) .

Embeddings exist in different forms. Word embeddings (Bengio, et al., 2003), Word2Vec (Mikolov, et al., 2013), Glove (Pennington, et al., 2014)), Character embeddings (Ma & Hovy, 2016) (Lample, et al., 2016) , Contextualized word embeddings ELMO (Peters, et al., 2017) (Peters, et al., 2018), BERT (Devlin, et al., 2018)), Contextualized character embeddings (Akbik, et al., 2018), etc.

3.4.4 RNN (Recurrent networks) and LSTM (Long Short term Memory)

Recurrent neural networks (RNN) use the idea of processing sequential information. In RNN output is fed back to the input. Therefore, RNNs have memory over previous computations and use this information in current processing. This property is naturally suited for many NLP tasks such as language modelling, machine translation, speech recognition, image captioning. This made RNNs increasingly popular for NLP applications in recent years (Young, et al., 2018).

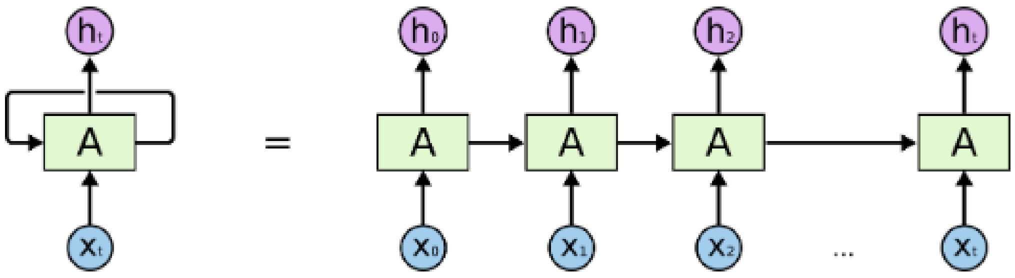


FIGURE 7. – RECURRENT NETWORK AND AN UNROLLED RECURRENT NEURAL NETWORK (SOURCE [50])

In order to overcome weaknesses of RNNs, such as vanishing and exploding gradient problems, new types of networks with gated units, such as LSTM (Long short term memory), GRUs (Gated Recurrent units), have been devised. Unlike vanilla RNN, LSTM and GRU allow the error to back-propagate through unlimited number of time steps (Young, et al., 2018).

[50] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

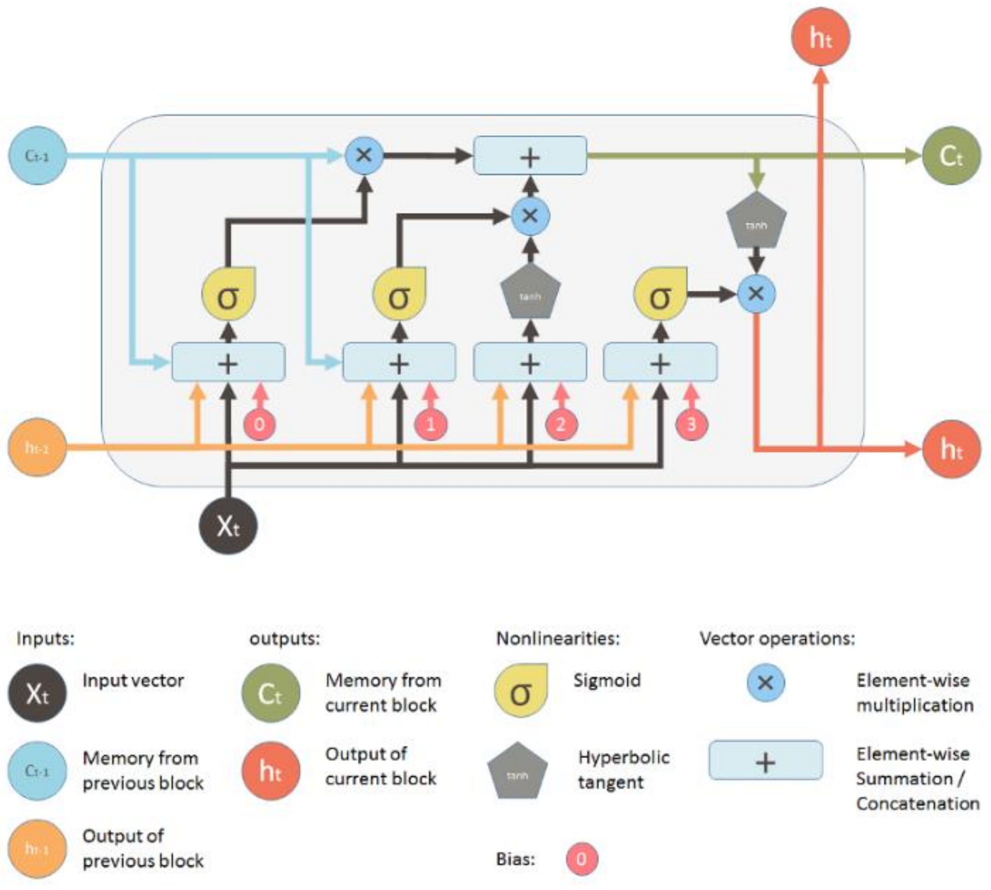


FIGURE 8. – LONG-SHORT TERM MEMORY BUILDING BLOCK (SOURCE: [51])

4. IMAGE METADATA EXTRACTION

In the monitoring of social media used in SoMeDi, we not only incorporate the analytics for available metadata such as text (see deliverable D3.2 for more details) but also incorporate the use of Artificial Intelligence to analyse images that are uploaded to the social media sites.

This is done using some of the latest advancements in neural networks, deep learning and computer vision. In this section we present a summary of the available technologies and our approach for the SoMeDi analysis pipeline.

4.1. Deep Learning

Convolutional neural networks (CNNs) are deep artificial neural networks that are used primarily to classify images (e.g. name what they see), cluster them by similarity (photo search), and perform object recognition within scenes. They are algorithms that can identify faces, individuals, street signs, medical objects of interest, kinds of animals, models of cars, handwriting and many other aspects of visual data. CNNs can also be applied to sound when

[51] https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

it is represented as a spectrogram. More recently, convolutional networks have been applied directly to text analytics as well as graph data with graph convolutional networks.

The efficacy of convolutional nets (ConvNets or CNNs) in image recognition is one of the main reasons of the machine learning boom going on in the past few years. They are currently powering major advances in computer vision (CV), which has obvious applications for self-driving cars, robotics, drones, security, medical diagnoses, and treatments for the visually impaired.

Using such approaches to analyse visual data such as the one found in images uploaded to the social media sites, we can extract metadata (e.g., recognition of features, language-oriented descriptors of images) that might be useful for the kind of analyses we target in SoMeDi. The machine learning paradigm is continuously evolving and new alternatives for problem resolution appear often monthly. An important factor is to match the developing machine learning models that run on the evolving hardware (e.g., GPU and CPU intensive machines such as the servers available in SoMeDi, but also ARM and heterogenous architectures found in mobile devices) so that applications are made smarter. Today, we have a myriad of frameworks at our disposal that allows us to develop tools that can offer a better level of abstraction along with the simplification of difficult programming challenges.

We will now briefly describe some available frameworks for Deep Learning and in section 4.2 we will propose our initial approach for using Deep learning in SoMeDi, to be further developed in new versions of this document. Each framework is built in a different manner for different purposes. Here, we will look at the top eight deep learning frameworks to give you a better idea of which framework will be the perfect fit or come handy in solving different business challenges.

4.1.1. TensorFlow

TensorFlow, developed in-house at Google originally for their own applications and then released for public usage, is arguably one of the best deep learning frameworks and has been adopted by many important customers and computer vision start-ups mainly due to its highly flexible system architecture. The most well-known use case of TensorFlow is Google Translate where it is used for capabilities such as natural language processing, text classification/summarization, speech/image/handwriting recognition, forecasting, and tagging.

TensorFlow is available on both desktop and mobile and also supports languages such as Python, C++, and R to create deep learning models along with wrapper libraries.

4.1.2. Caffe

Caffe is a deep learning framework that is supported with interfaces like C, C++, Python, and MATLAB as well as the command line interface. It is well known for its speed and portability and its applicability in modelling convolution neural networks (CNN). The biggest benefit of using Caffe's C++ library (comes with a Python interface) is the ability to access off-the-shelf available networks from the deep net repository Caffe Model Zoo that are pre-trained and can be used immediately. When it comes to modeling CNNs or solving image processing issues, this should be your go-to library.

Caffe's biggest asset is speed. It can process over 60 million images on a daily basis with a single Nvidia K40 GPU. That's 1 ms/image for inference and 4 ms/image for learning — and more recent library versions are faster still. Caffe is a popular deep learning network for visual recognition. However, Caffe does not support fine-granular network layers like those found in TensorFlow. Given the architecture, the overall support for recurrent networks, and language modelling are quite poor, and establishing complex layer types has to be done in a low-level language.

4.1.3.Torch (and pyTorch)

Torch is a scientific computing framework that offers wide support for machine learning algorithms. It is a Lua-based deep learning framework and is used widely amongst industry giants such as Facebook, Twitter, and Google. It employs CUDA along with C/C++ libraries for processing and was basically made to scale the production of building models and to provide overall flexibility.

As of now, PyTorch has seen a high level of adoption within the deep learning framework community and is considered to be a competitor to TensorFlow. PyTorch is basically a Python port to the Torch deep learning framework used for constructing deep neural networks and executing tensor computations that are high in terms of complexity. The usage of Python removes one of the hurdles and limitations of Torch (using the Lua scripting language, which is not as expressive or widespread), which means that anyone with a basic understanding of Python can get started on building their own deep learning models. Given PyTorch framework’s architectural style, the entire deep modelling process is far simpler as well as transparent compared to Torch.

4.1.3.DeepLearning4j

Parallel training through iterative reduce, microservice architecture adaptation, and distributed CPUs and GPUs are some of the salient features of the Deeplearning4j (DL4J) deep learning framework. It is developed in Java as well as in Scala (a JVM-based scripting language) and supports other JVM languages, too. Widely adopted as a commercial, industry-focused distributed deep learning platform, the biggest advantage of this deep learning framework is that it can bring together the entire Java ecosystem to execute deep learning. It can also be administered on top of Hadoop and Spark to orchestrate multiple host threads. DL4J uses MapReduce to train the network while depending on other libraries to execute large matrix operations.

Deeplearning4j comes with a deep network support through RBM, DBN, convolution neural networks (CNNs), recurrent neural networks (RNNs), recursive neural tensor networks (RNTNs), and long short-term memory (LTSM).

Since this deep learning framework is implemented in Java, it is much more efficient compared to Python. When it comes to image recognition tasks using multiple GPUs, it is as fast as Caffe. This framework shows matchless potential for image recognition, fraud detection, text mining, parts-of-speech tagging, and natural language processing.

5. Conclusions

In this document we have provided more State of the Art for the topics of interest in project SoMeDi, such as data extraction, NLP elements and image metadata extraction. These technologies are currently either working already in our v1 prototypes or making their way into the designs for v2 technology to be delivered by project's end.

The work in this deliverable version 1 was selected by ITEA for inclusion in its Living Roadmap for Smart Communities. This signals that the information that we collect in the project is seen as useful by the larger software research community around ITEA and EUREKA. We expect that the new results in v2 are deemed equally significant. Findings about the consortium work in studying and integrating State of the Art are also disseminated throughout the communication channels of the project such as social media and scientific publications.

The next steps for SoMeDi in this regard are to maintain the vigilance on the State of the Art throughout the remaining final year of the project. In that regard, the key achievement to deliver during the final phase will be deliverable **D1.4 SoMeDi Outlook** which will summarize the findings in the project while putting them into perspective in the moving spectrum of the State of the Art. This deliverable will be released in the final months of the project.

BIBLIOGRAPHY

Lample, G. et al., 2016. Neural Architectures for Named Entity Recognition. *arXiv:1603.01360.*

Akbik, A., Blythe, D. & Vollgraf, R., 2018. Contextual String Embeddings for Sequence Labeling. *27th International Conference on Computational Linguistics.*

Bengio, Y., Ducharme, R., Vincent, P. & Jauvin, C., 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research,* p. 1137–1155.

Devlin, J., Chang, M., Lee, K. & Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805.*

Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep learning.* s.l.:MIT Press.

Ma, X. & Hovy, E., 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *arXiv:1603.01354.*

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013. *Efficient Estimation of Word Representations in Vector Space.* s.l.:arXiv:1301.3781.

Mikolov, T. et al., 2010. *Recurrent neural network based language model.* s.l.:Interspeech.

Pennington, J., Socher, R. & Manning, C. D., 2014. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* Volume 1532–1543.

Peters, M. E., Ammar, W., Bhagavatula, C. & Power, R., 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv:1705.00108.*

Peters, M. E. et al., 2018. Deep contextualized word representations. *Proc. of NAACL.*

Socher, R. et al., 2013. *Recursive deep models for semantic compositionality over a sentiment treebank.* s.l.:Proceedings of the conference on empirical methods in nautral language processing (EMNLP).

Young, T., Hazarika, D., Poria, S. & Cambria, E., 2018. Recent Trends in Deep Learning Based Natural Language Processing. *arXiv:1708.02709.*