|  |  |  |
| --- | --- | --- |
| ***Delivery Date*:**  M18 31/05/2018 | ***Project Number*:**  ITEA3 Call2 15011 | ***Responsible partner*:** |



*D3.2 DII Text Intelligence Toolkit*

*D3.3* DII Metadata mining and Model based techniques

*WP3 - Digital Interaction Intelligence techniques* *– T3.3. DII Metadata mining and model based techniques*

Vision, architecture and data integration

Document Contributors

|  |  |  |
| --- | --- | --- |
| Name | Company | Email |
| George Suciu | BEIA | [george@beia.ro](mailto:george@beia.ro) |
| Elena Muelas Cano | HI Iberia Ingeniería y Proyectos | [emuelas@hi-iberia.es](mailto:emuelas@hi-iberia.es) |
| Raúl Santos de la Cámara | HI Iberia Ingeniería y Proyectos | [rsantos@hi-iberia.es](mailto:rsantos@hi-iberia.es) |
| Yihwa Kim | Taiger | [yihwa.kim@taiger.com](mailto:yihwa.kim@taiger.com) |
|  |  |  |
|  |  |  |

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 with D3.2 contents, new text for image processing analytics, updated content for Text analysis in section 3. |
| 0.91 |  | HIB |  |
| 0.92 | 15/10/2018 | TAIGER |  |
| 0.93 | 23/10/2018 | BEIA | Revised version |
|  | 17/12/2018 | Taiger | Adding Deep learning based NLP |

Table of contents

[Document Contributors 2](#_Toc532825979)

[Document History 2](#_Toc532825980)

[1. Introduction 5](#_Toc532825981)

[2. Data Extraction 6](#_Toc532825982)

[2.1. Data Extraction Techniques [Marketing Use Case] 6](#_Toc532825983)

[2.2. Data Extraction Techniques [Recruiting Use Case] 6](#_Toc532825984)

[2.3. Data Extraction Techniques [Turkish Use Case] 7](#_Toc532825985)

[2.4. Data Models 7](#_Toc532825986)

[Data Models 7](#_Toc532825987)

[3. Object Recognition 8](#_Toc532825988)

[3.1. Resources used 8](#_Toc532825989)

[3.2. Proposed Pipeline 8](#_Toc532825990)

[3.3. Proposed architecture for SoMeDi 9](#_Toc532825991)

[4. DII Text Intelligence tookit for Marketing use case 11](#_Toc532825992)

[4.1 Architecture 11](#_Toc532825993)

[4.2 Architecture for training NER (Named Entity Recognition) and sentiment classifier 11](#_Toc532825994)

[4.2.1 NER (Named Entity Recognition) 12](#_Toc532825995)

[4.2.2 Sentiment analysis 12](#_Toc532825996)

[4.3 Deployment 15](#_Toc532825997)

[5. DII Text Intelligence Toolkit 17](#_Toc532825998)

[5.1. Recruitment Scenario Description 17](#_Toc532825999)

[5.2. Methods used for Sentiment Analysis 17](#_Toc532826000)

[Sentiment Analysis is part of the Text Analytics. 17](#_Toc532826001)

[Methods based on Machine Learning 17](#_Toc532826002)

[5.3. Description of the Microsoft Azure Cognitive Services – Text Analytics Project 18](#_Toc532826003)

[5.4. Description of the Stanford CoreNLP Sentiment Analysis Project 22](#_Toc532826004)

[5.5. Software Development 23](#_Toc532826005)

[5.6. Integration with the SoMeDi platform 23](#_Toc532826006)

[Overview 23](#_Toc532826007)

[API contract 24](#_Toc532826008)

[Security 25](#_Toc532826009)

[Compliance 25](#_Toc532826010)

[Telemetry 25](#_Toc532826011)

[6. Aligning metadata intelligence with recruitment use case 27](#_Toc532826012)

[7. Aligning metadata intelligence with marketing use case 28](#_Toc532826013)

[8. Aligning metadata intelligence with NBA use case 29](#_Toc532826014)

[8.Conclusions 30](#_Toc532826015)

[References 31](#_Toc532826016)

# **Introduction**

The management of complex heterogeneous data requires the selection of suitable mining methods as well as of appropriate modelling techniques. The focus of task T3.3 is set on how to handle and store the different type of available data in order to improve the DII.

The overall objective of Task 3.3 is to provide a dynamic document that describes the management of complex heterogeneous data regarding the SoMeDi platform functionalities – with the two proposed use cases for marketing and recruiting.

Deliverable D3.3 requires the selection of suitable mining methods as well as of appropriate modeling techniques. The focus of this task will concern the analysis of data extraction techniques ideal for NLP solutions (Romanian partners use case), but also data mining approaches for social media concerning the marketing use case.

The material presented resulted from the work performed by the responsible partners: this implies a literature review and 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’s DID toolkit. The field research was conducted in the above-mentioned domains: data extraction, natural language processing, and opinion mining.

This first iteration of the D3.3 document is organized as follows:

* Section 2 describes the concept of data extraction concerning SoMeDi platform functionalities, setting an overview of the possibilities to acquire data from the internet;
* Section 3 presents the NLP techniques and service frameworks for extracting information; also, in this section, we set an overview for specific algorithms desired to be implemented in the Romanian use case, the purpose of these algorithms is to assure the matchmaking between the internship candidates skills and company expectations;
* Section 4 provides the general outline of the DII component for the extraction of metadata from images in social media.
* Section 5 describes metadata mining solutions engaged in the marketing use case;
* Section 6 concludes the document.

|  |
| --- |
| **WP3 - Digital Interaction Intelligence techniques (TAIGER)** |
| T3.1. DII Software SOTA and guidance (SIVECO) |
| T3.2. DII Text intelligence (HIB) |
| T3.3. DII Metadata mining and model based techniques (BEIA) |
| D3.1. State-of-the-art and guidance Report (SIVECO) - Doc |
| D3.2. DII Text intelligence Toolkit - (HIB) - SW |
| D3.3. DII Metadata intelligence and Model based techniques (BEIA) - SW |

# **2. Data Extraction**

Data extraction is the act or process of retrieving [data](https://en.wikipedia.org/wiki/Data" \o "Data) out of (usually [unstructured](https://en.wikipedia.org/wiki/Unstructured_data" \o "Unstructured data) or poorly structured) data sources for further [data processing](https://en.wikipedia.org/wiki/Data_processing" \o "Data processing) or [data storage](https://en.wikipedia.org/wiki/Data_storage_device" \o "Data storage device) ([data migration](https://en.wikipedia.org/wiki/Data_migration" \o "Data migration)). The [import](https://en.wikipedia.org/wiki/Data_import" \o "Data import) into the intermediate extracting system is thus usually followed by [data transformation](https://en.wikipedia.org/wiki/Data_transformation" \o "Data transformation) and possibly the addition of [metadata](https://en.wikipedia.org/wiki/Metadata" \o "Metadata) prior to [export](https://en.wikipedia.org/wiki/Data_export" \o "Data export)ing to another stage in the data [workflow](https://en.wikipedia.org/wiki/Workflow" \o "Workflow).

In the following sections are presented the data extraction methods applied for each of SOMEDI use cases.

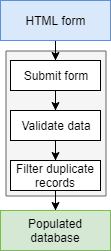
## 2.1. Data Extraction Techniques [Marketing Use Case]

In order to acquire content from the internet ….

## 2.2. Data Extraction Techniques [Recruiting Use Case]

Data collection from the end users was realized by using several Web Forms who write data to a SQL database on a SQL server.

The information gathered thorught the HTML forms is later handed to a downstream data extraction process. The HTML form invokes a Common Gateway interaction (CGI) request to the Web server.



The two methods to submit a form for CGI processing are presented below:

* First method, using the HTTP POST verb, the HTML forms can be submitted with (name, value) pairs encoded in the body of the request.
* Second method, using the HTTP GET verb, forms can be submitted by supplying (name, value) pairs in the URL.

The SoMeDi platform includes the option to secure communication and submissions of HTML forms through website by using Secure Hyper Text Transfer Protocol or https.

Submissions through the secured web form are stored in a way that only authorized and authenticated users can view the results

## 2.3. Data Extraction Techniques [Turkish Use Case]

…….

## 2.4. Data Models

Data Warehouse

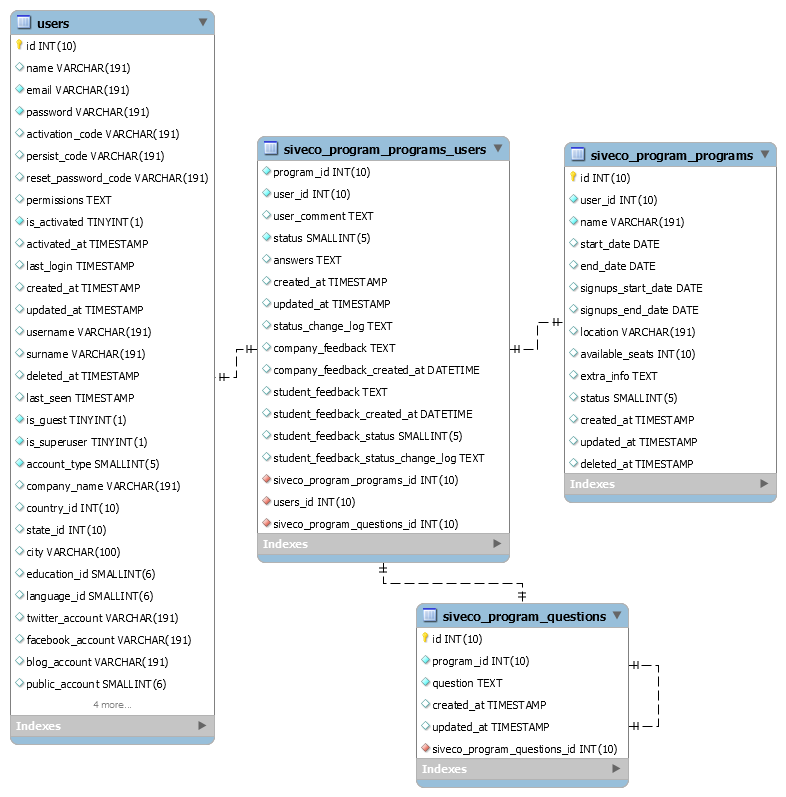
For the Somedi recruiting use case InnoDB engine is used, which supports foreign key and transaction. The default character set for this table is UTF8, which supports all languages for internationalization. The Data store repository with complete view of the business data is as follows:

* Aggregated data from multiple sources
* Active users, applicants, companies
* Programs

### Data Models

**More here ///<https://www.studytonight.com/dbms/database-model.php>**

The Somedi Recruitment platform is built on October CMS. October CMS provides a simple Active Record implementation for working with the database environment, based on [Eloquent by Laravel](http://laravel.com/docs/eloquent). Each database table has a corresponding "Model" which is used to interact with that table. Models allow to query for data in all tables, as well as insert new records into the table.





















# **3. Object Recognition**

For object recognition our target use case is the description of elements that can be found in images that users upload to their social media accounts (Twitter, Facebook, etc.). The result of these analyses are output as text metadata that can be incorporated to the index that is generated for the rest of the components of the message (textual information, geopositioning, etc.).

For the first iteration of the technology in SoMeDi we have used networks that describe generic object recognition, that is, the results are general purpose and not domain-specific. In future iterations (D3.3 version 2) we will investigate our options to include such domain specific networks, such as ones trained to describe restaurant elements (for the marketing use case).

## 3.1. Resources used

RESNET won the 1st places in: ImageNet classification, ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation. ILSVRC object recognition challenges (2015)[[1]](#footnote-46).

DENSECAP gets speed and accuracy improvements over baselines based on current state of the art. The model generates rich snippet descriptions of regions and accurately grounds the captions in the images.

## 3.2. Proposed Pipeline

We propose for SoMeDi the usage of the following resources which we will now examine in detail:

* Deep Learning Models
* Deep Learning Frameworks: Torch
* Objects Datasets: Imagenet[[2]](#footnote-47), VisualGenome[[3]](#footnote-48), MS-COCO[[4]](#footnote-49)

We mount two DL meta-architectures working in parallel. These DL meta-architectures, ResNet and DenseCap, are specialized in different tasks.

**A: ResNet**[[5]](#footnote-50) models is trained in 1000 labels (objects) using ImageNet dataset. ResNet models get very accurate results in object recognition tasks. These models have very good relation accuracy vs. speed. ResNet models implement several levels of deep layers [18, 34, 50, 101, 152]. ResNet models with more layers will get more accuracy, but these models will be slower. ResNet architecture uses a deep residual learning block to address the degradation problem with very deep networks.

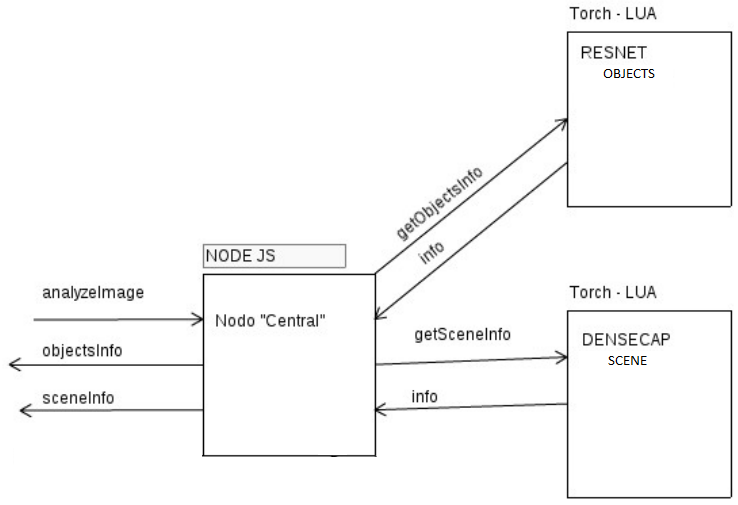
Image (224x224) validation error rate

|  |  |  |
| --- | --- | --- |
| Network | Top-1 error | Top-5 error |
| ResNet-18 | 30.43 | 10.76 |
| ResNet-34 | 26.73 | 8.74 |
| ResNet-50 | 24.01 | 7.02 |
| ResNet-101 | 22.44 | 6.21 |
| ResNet-152 | 22.16 | 6.16 |
| ResNet-200 | 21.66 | 5.79 |

**B: DenseCap**[[6]](#footnote-51) model task is to describe images in natural language. DenseCap identifies isolated objects and group of objects like one entity. DenseCap is trained/validated on the Visual Genome[[7]](#footnote-52) dataset, which comprises 94,000 images and 4,100,000 region-grounded captions. The architecture is composed of a CNN[[8]](#footnote-53), a dense localization layer and an RNN[[9]](#footnote-54)

## 3.3. Proposed architecture for SoMeDi

The proposed architecture for the image metadata extraction DII module is as follows:



As we can see, internally it is comprised of two distinct analysis modules. One is based on the RESNET network and is used for recognition of objects of interest and the other is based on the DENSECAP network and is used to provide scene description of images.

As it can be seen in the Figure, we have divided the analysis of the images into 4 parts:

\* The "central" node, which receives the messages (images scraped from the Social Media) to analyze and is responsible for distributing according to the type of analysis requested by the source.

The analysis nodes (according to the information to be obtained from each frame): RESNET [objects in the image], DENSECAP [global description of the image and the different parts of it]. The "central" node is written in Node.js, and only performs message forwarding functions of both the part of middleware towards deep learning, as well as the part of deep learning towards middleware. The RESNET part uses the Torch library to perform all the analysis and is written in LUA. For each message it receives, it generates a list of objects that appear in the image, ordering this list according to the preponderance of the same in the scene. We always show the 5 first, but this value can be changed to show some more, although in this case it would increase the number of mistakes and errors in the results.

The DENSECAP network, like RESNET, uses Torch and is written in LUA. With it we get descriptions of the scenes. This network divides the image into as many pieces as indicated (we usually work with 50) and analyzes the image in a generic way and each one of those (50) sub-images to give a description.

The languages and frameworks used for this DII module are as follow:

* For the Deep Learning modules we use Torch[[10]](#footnote-55) which is usually programmed using the LUA scripting language. However, Lua is a language of limited flexibility and cross project support, so as of the writing of this document we are migrating the system to PyTorch[[11]](#footnote-56) which is a compatible implementation with a Python front-end. This enables us as well to use similarly minded languages and approaches compared to the ones used in the text analytics (see D3.2).
* For general purpose aspects in the DII module’s API facing side (message reception and queueing, input-output analysis using REST APIs) we use a custom module programmed in Node.js[[12]](#footnote-57). This is a very efficient server side Javascript based architecture for applications that perform processing on large data streams such as SoMeDi.

With the help of these components, our system runs quite efficiently in our test servers with ample headroom to accommodate connection to intense streams coming from social media for the Use Cases of the project.

|  |  |  |
| --- | --- | --- |
| RESNET |  |  |
|  | Nvidia Hardware used | Throughput |
|  | 980 | 18 fps |
|  | 1080 | 28 fps |
|  |  |  |
| DENSECAP |  |  |
|  | Nvidia Hardware used | Throughput |
|  | 980 | 7 fps |
|  | 1080 | 10 fps |

The reference frames used are 1080p (~2 megapixel) resolution images (1920 \* 1080 pixels) which are a compromise between performance and the accuracy of the results.

# 4. DII Text Intelligence tookit for Marketing use case

## 4.1 Architecture

We have used Python3 with conda[[13]](#footnote-58) and used PyTorch[[14]](#footnote-59) for deep learning framework.

Recently there have been big advances in Natural Language Processing with deep learning, and lots of open sourced tools are available: to name a few those from allenNLP[[15]](#footnote-60) (ELMO). Zalando Research (Flair[[16]](#footnote-61)), FastAI[[17]](#footnote-62) and Google (BERT[[18]](#footnote-63)). We have used Flair, due to its simplicity in its use, and its light architecture as well as due to the fact that it showed state of the art results for a couple of NLP tasks (as of 2018 November, NER English , NER German, Chunking and PoS tagging).

Flair (Akbik, et al., 2018) implements contextualized character-level word embeddings which combine the best attributes of the previously existed embeddings: the ability to (1) pre-train on large unlabeled corpora, (2) capture word meaning in context and therefore produce different embeddings for polysemous words depending on their usage, and (3) model words and context fundamentally as sequences of characters, to both better handle rare and misspelled words as well as model subword structures such as prefixes and endings (Akbik, et al., 2018).

Character level contextual embeddings are based on neural language modeling (LM) that have allowed language to be modeled as distributions over sequences of characters instead of words (Sutskever, et al., 2014; Graves, 2013; Kim, et al., 2015). Recent work has shown that by learning to predict the next character on the basis of previous characters, such models learn internal representations that capture syntactic and semantic properties: even though trained without an explicit notion of word and sentence boundaries, they have been shown to generate grammatically correct text, including words, subclauses, quotes and sentences (Sutskever, et al., 2014; Graves, 2013; Karpathy, et al., 2015). More recently, Radford and colleagues (Radford, et al., 2017) showed that individual neurons in a large LSTM-LM can be attributed to specific semantic functions, such as predicting sentiment, without explicitly trained on a sentiment label set (Akbik, et al., 2018).

## 4.2 Architecture for training NER (Named Entity Recognition) and sentiment classifier

In SoMeDi, we have implemented English and Spanish Named Entity Recognizer as well as Sentiment Analyzer. For this we have trained Spanish character based context aware language model on the GPU machine on AWS (p2.xlarge, P2 Tesla K-series K-80) with 61 GB of memory and 12 GB of GPU memory for 2 weeks (each model).

Trained character based language model was used for Spanish NER and Spanish Sentiment analyser, as part of embedding. For Spanish NER, and Spanish and English sentiment analyser, we have trained each of them about one day.

### 4.2.1 NER (Named Entity Recognition)

Named Entity Recognition is a task where entities such as Person, Organization, Location are extracted from the unlabeled sentences.

We have used CONLL2002 dataset[[19]](#footnote-64) to train Spanish NER. We have acquired F1 value of 85.92 for validation set, and 87.58 for the test set, which is higher than the previous state of the art of Spanish NER (85.77 by (Yang, et al., 2017)) .

For English NER, we have used existing implementation of Flair. This model is current state of the art model with F1 value of 93.09 (Akbik, et al., 2018).

### 4.2.2 Sentiment analysis

Opinions are central to almost all human activities because they are key influencers of our behaviors. With the explosive growth of social media (e.g. reviews, forum discussions, blogs, micro-blogs, Twitter, comments, and postings in social network sites) on the Web, individuals and organizations are increasingly using the content in these media for decision making. Sentiment analysis, also called opinion mining, is the field of study that analyzes people’s opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products services, organizations, individuals, issues, evens, topics, and their attributes. (Liu, 2012)

We have trained English sentiment analyzer with a dataset[[20]](#footnote-65) which contains 1,578,627 classified tweets. We have achieved F1 value of .816 with this dataset. For Spanish sentiment analyzer, we have combined datasets from TASS2012[[21]](#footnote-66) and TASS2018[[22]](#footnote-67), which in total contained 10,026 classified tweets. We have reached F1 value .4763 with this dataset.

# tag type for prediction

tag\_type = 'ner'

# making tag dictionary from the corpus

tag\_dictionary = corpus.make\_tag\_dictionary(tag\_type=tag\_type)

print(tag\_dictionary.idx2item)

# initialize embeddings

embedding\_types: List[TokenEmbeddings] = [

WordEmbeddings('es-glove'),

CharLMEmbeddings('./resources/taggers/language\_model\_es\_forward\_long/best-lm.pt'),

CharLMEmbeddings('./resources/taggers/language\_model\_es\_backward\_long/best-lm.pt'),

]

embeddings: StackedEmbeddings = StackedEmbeddings(embeddings=embedding\_types)

# initialize sequence tagger

from flair.models import SequenceTagger

tagger: SequenceTagger = SequenceTagger(hidden\_size=128,

embeddings=embeddings,

tag\_dictionary=tag\_dictionary,

tag\_type=tag\_type,

use\_crf=True)

# initialize trainer

from flair.trainers.sequence\_tagger\_trainer import SequenceTaggerTrainer

trainer: SequenceTaggerTrainer = SequenceTaggerTrainer(tagger, corpus, test\_mode=False)

#train

trainer.train('resources/taggers/es-ner-long-glove', learning\_rate=0.1, mini\_batch\_size=14, max\_epochs=150, patience = 4 )

Part of Code for training Spanish NER.

# making a list of word embeddings

word\_embeddings = [WordEmbeddings('en-twitter-glove'),

CharLMEmbeddings('mix-forward'),

CharLMEmbeddings('mix-backward')]

# initialize document embedding by passing list of word embeddings

document\_embeddings: DocumentLSTMEmbeddings = DocumentLSTMEmbeddings(word\_embeddings,

hidden\_states=512,

reproject\_words=True,

reproject\_words\_dimension=256,)

# create text classifier

classifier = TextClassifier(document\_embeddings, label\_dictionary=label\_dict, multi\_label=False)

#initialize text classifier trainer

trainer = TextClassifierTrainer(classifier, corpus, label\_dict)

# start the training

trainer.train('resources/sentiment\_classifier-en-3classes-nookandsemeval2013/results',

learning\_rate=0.1,

mini\_batch\_size=32,

anneal\_factor=0.5,

patience=5,

max\_epochs=150)

Part of code for training English sentiment classifier

## 4.3 Deployment

Trained NER and sentiment classifier were deployed using Flask68 and Docker69.

Flask[[23]](#footnote-68) is a micro web framework for Python, for developing webapps. Docker[[24]](#footnote-69) performs operating-system-level virtualization, also known as ‘containerization’. Containers are created from ‘images’ that specify their precise contents. Docker can package an application and its dependencies in a virtual container that can run on any Linux server. This helps enable flexibility and portability on where the application can run[[25]](#footnote-70).

FROM ubuntu:latest

MAINTAINER Yihwa Kim "yihwa.kim@taiger.com"

RUN apt-get clean -y

RUN apt-get update -y

RUN apt-get install -y python3-pip python3-dev build-essential git-core

COPY . /app

WORKDIR /app

ENV LC\_ALL=C.UTF-8

ENV LANG=C.UTF-8

RUN git clone https://github.com/zalandoresearch/flair.git

RUN pip3 install -r requirements.txt

#ENTRYPOINT ["python3"]

ENV FLASK\_APP=ner-flair-predict.py

#ADD ./model/es-ner-glove.pt /app/flair/model/es-ner-glove.pt

EXPOSE 5000

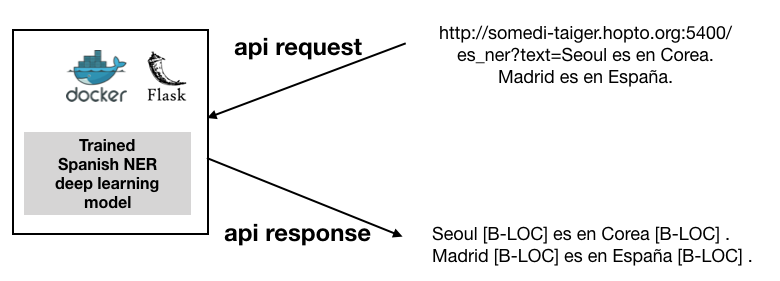
ENTRYPOINT ["./entrypoint.sh"]

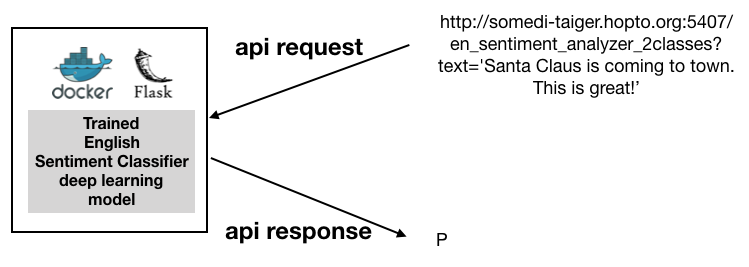
Content of Dockerfile

docker build -t image-ner-es:latest .

docker run -itd -p 5400:5000 --name ner-es image-ner-es

Commands to create docker image and run docker container from the image





Schematics of NER (Spanish) and sentiment classifier (English)

# 5. DII Text Intelligence tookit for Recruiting use case

## 5.1. Recruitment Scenario Description

This use case applies Sentiment Analysis (SA) techniques to improve the recruitment processes aiming to increase the efficiency of internship campaigns by ensuring a better match between the candidates' professional skills and the hiring company fields of activity.

Therefore, by accessing the SoMeDi platform, the candidates will complete their profile data and further - browse, select, and apply to specific internship programmes. The application process implies that the internship candidates will complete several forms providing feedback regarding the companies fields of activity.

The NLP tool advanced in this project analyzes each field completed, containing the text written by the candidate. The application delivers a score (that approximates how interested the candidate is to work in each area) namely a value between 0 and 1. If the sentiment analysis score is close to 0 means that there is no interest, while a value close to 1 signifies that the candidate is interested.

As mentioned earlier, the sentence analysis is performed by using NLP Text Analytics, namely Sentiment Analysis. The current version of the DII tool is delivered in this phase of the project by deploying the following methods (services) for sentiment analysis:

a) the first method uses services from Microsoft Azure - Cognitive Services;

b) the second method is built with open source Stanford CoreNLP.

We developed a sentiment analysis application using an open source code written in C # to present an appropriate GUI for the internship application. The sentiment analysis application has versions in English and Romanian, and for analyzing Romanian language content (text input) was used a Romanian-English translation service from MS Azure Translator Text.

In the following sections we will present the methods used for delveoping the sentiment analysis applications, describe each of the SA solutions (Azure and Stanford CoreNLP), and also release the instructions on how to use/test the above-mentioned applications.

## 5.2. Methods used for Sentiment Analysis

#### Sentiment Analysis is part of the Text Analytics.

Understanding and analyzing unstructured text is an increasingly popular field and includes a broad spectrum of problems such as sentiment analysis, key phrase extraction, topic modeling/extraction, aspect extraction and more. A simple approach is to keep a lexicon of words or phrases that assess negative or positive sentiment to a sentence (e.g., the words “bad”, “hate”, “not good” would belong to the lexicon of negative words, while “good”, “great”, “like” would belong to the lexicon of positive words). But this means such lexicons must be manually curated, and even then, they are not always accurate.

#### Methods based on Machine Learning

A more robust approach is to train models that detect sentiment. Here is how the training process works – a large dataset of text records is created that was already labeled with sentiment for each record. The first step is to tokenize the input text into individual words, then apply stemming[[26]](#footnote-71) (stemming is the process of reducing inflected -or sometimes derived- words to their base or root form). Next, it is necessary to construct features from these words; these features are used to train a classifier. Upon completion of the training process, the classifier can be used to predict the sentiment of any new piece of text. It is essential to construct meaningful features for the classifier, and the list of features includes several from state-of-the-art research:

* N-grams[[27]](#footnote-72) denote all occurrences of *n* consecutive words in the input text. The precise value of *n* may vary across scenarios, but it’s common to pick *n=2* or *n=3;*
* Part-of-speech tagging [[28]](#footnote-73) [[29]](#footnote-74) is the process of assigning a part-of-speech to each word in the input text;
* Word embeddings[[30]](#footnote-75) [[31]](#footnote-76) are a recent development in natural language processing, where words or phrases that are syntactically similar are mapped closer together. Neural networks are a popular choice for constructing such a mapping. For sentiment analysis, neural networks that encode the associated sentiment information are used as well. The layers of the neural network are then used as features for the classifier.

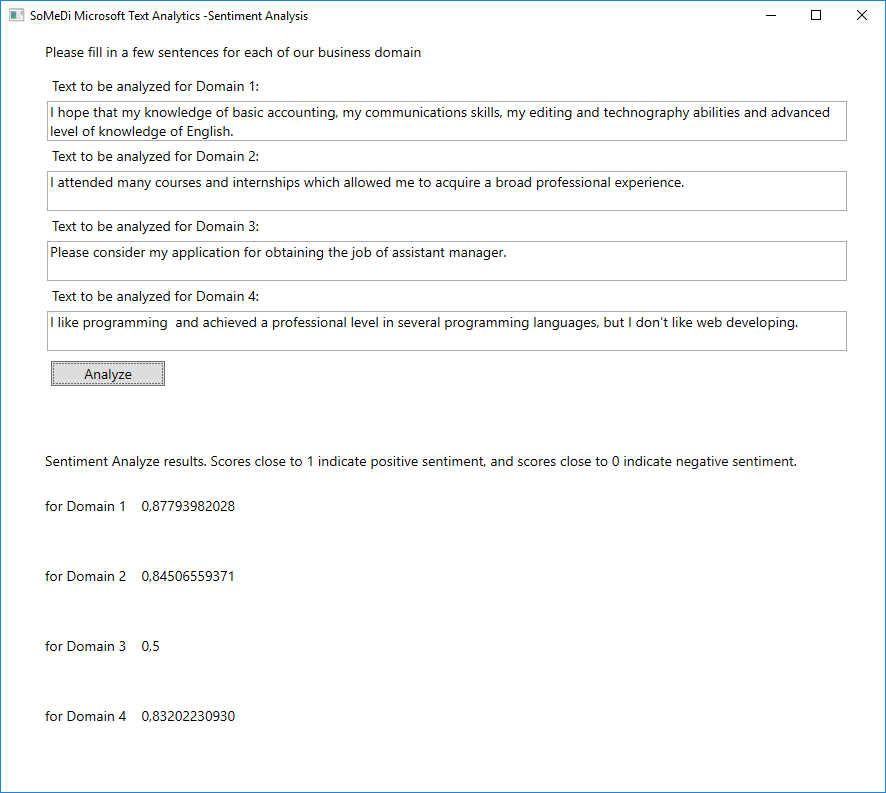
## 5.3. Description of the Microsoft Azure Cognitive Services – Text Analytics Project

Text Analytics uses a machine learning classification algorithm to generate a sentiment score between 0 and 1. Scores closer to 1 indicate positive sentiment, while scores closer to 0 indicate negative sentiment.

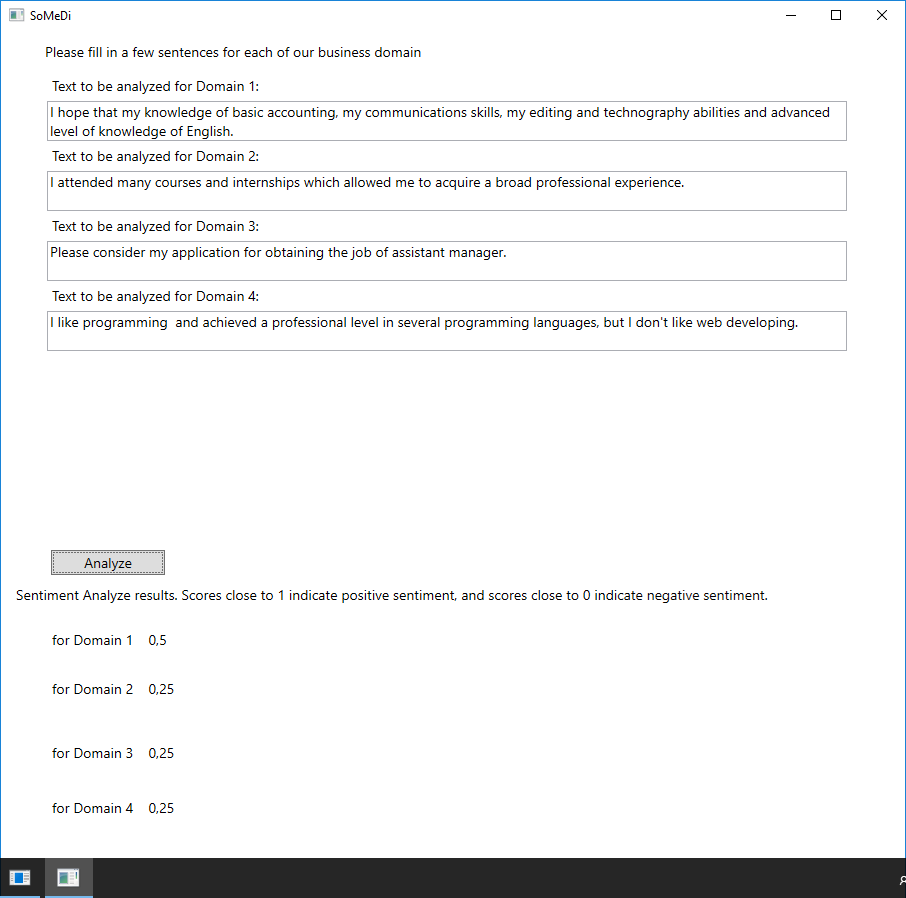
The model is pretrained with an extensive body of text with sentiment associations. Currently, it is not possible to provide your own training data. No labeled or training data is needed to use the service.

The model uses a combination of techniques during text analysis, including text processing, part-of-speech analysis, word placement, and word associations.

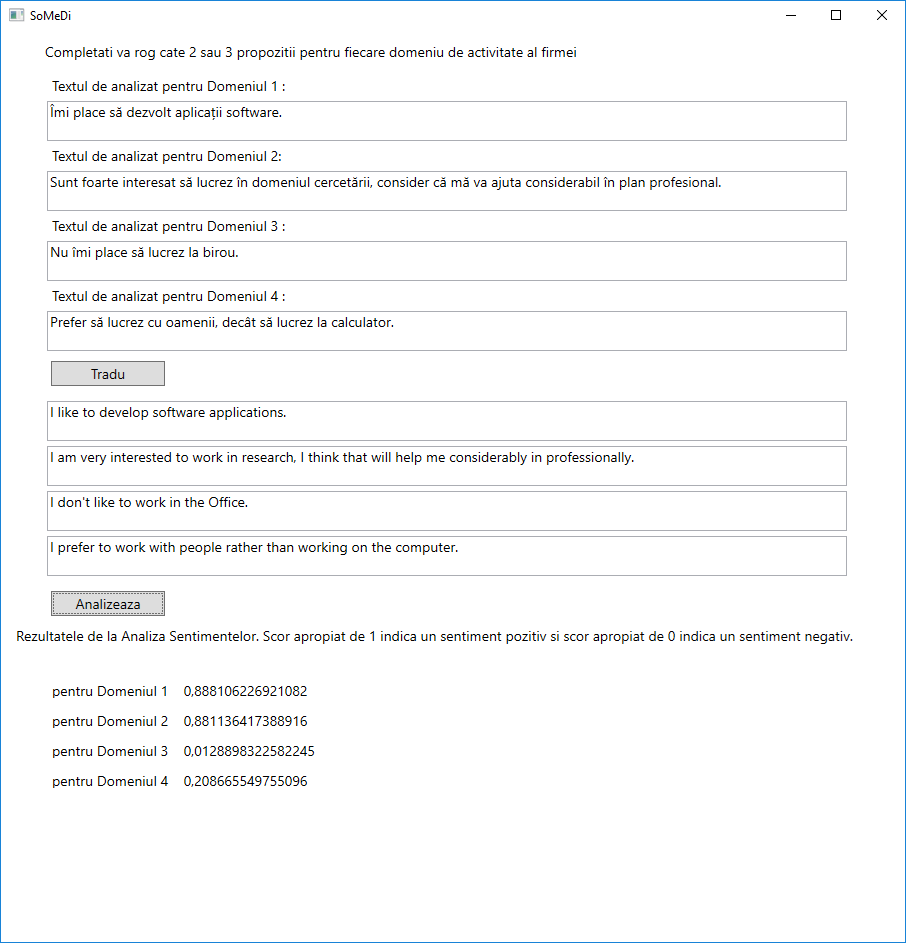
Sentiment analysis is performed on the entire document, as opposed to extracting sentiment for a particular entity in the text. In practice, there is a tendency for scoring accuracy to improve when documents contain one or two sentences rather than a large block of text. During an objectivity assessment phase, the model determines whether a document as a whole is objective or contains sentiment. A document that is mostly objective does not progress to the sentiment detection phrase, resulting in a 0.50 score, with no further processing. For documents continuing in the pipeline, the next phase generates a score above or below 0.50, depending on the degree of sentiment detected in the document.



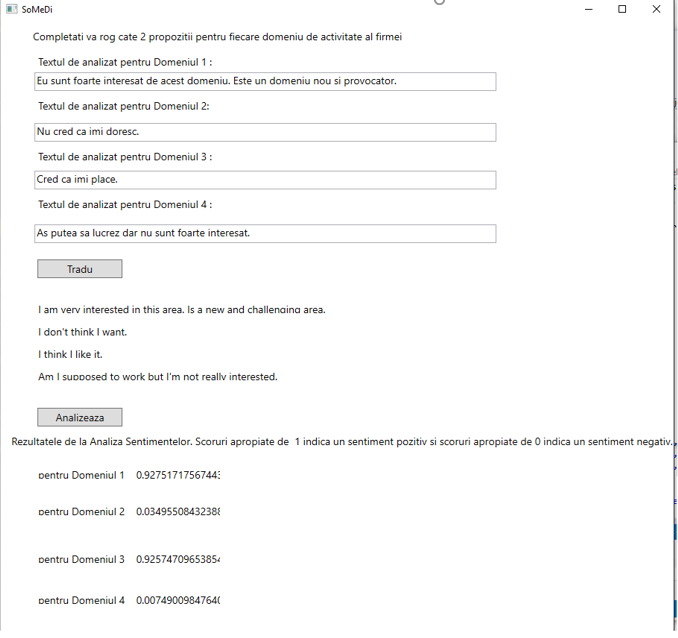
(a) Microsoft Azure Sentiment Analysis application (EN)



(b) StanfordNLP Sentiment Analysis application (EN)



(c) Microsoft Azure Sentiment Analysis application (RO)



(d) StanfordNLP Sentiment Analysis application (RO)

## 5.4. Description of the Stanford CoreNLP Sentiment Analysis Project

Stanford CoreNLP provides a set of human language technology tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences regarding phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get the quotes people said, etc.

The following Stanford CoreNLP tools are necessary for this project for Sentiment Analysis, in order to: tokenize, ssplit, pos, lemma, parse, sentiment; ner and dcoref are not necessary.

The functions of these tools are:

1. tokenize: Tokenizes the text into a sequence of tokens,
2. ssplit: Splits a sequence of tokens into sentences,
3. pos: Labels tokens with their part-of-speech (POS) tag,
4. lemma: Generates the lemmas (base forms) for all tokens; including Sentiment Class,
5. parse: Provides full syntactic analysis, including, both constituent and dependency representation tokens,
6. sentiment: Sentiment analysis with a compositional model over trees using deep learning.

Stanford CoreNLP is written in Java. Stanford CoreNLP introduced two new ideas: a) the Stanford Sentiment Treebank and b) a powerful Recursive Neural Tensor Network.

A treebank can be defined as a linguistically annotated corpus (Data Base) that includes some grammatical analysis beyond the part-of-speech level. The Stanford Sentiment Treebank is the first corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. It includes labels for every syntactically plausible phrase in thousands of sentences.

Recursive Neural Tensor Networks (RNTN) take as input phrases of any length. They represent a phrase through word vectors and a parse tree and then compute vectors for higher nodes in the tree using the same tensor-based composition function. It is placed on top of grammatical structures. A phrase is composed of a couple of meaning related words/tokens. The tri-gram is used.

The Deep learning model builds up a representation of the whole sentence based on the sentence structure. It computes the sentiment based on how words compose the meaning of longer phases. The activation function is: a) f=tanh() for hidden layers and b) f= softmax() for output layer used for 5-class classification. These 5-class sentiment classifications are: VERY NEGATIVE, NEGATIVE, NEUTRAL, POSITIVE, VERY POSITIVE.

Training of RNTN is performed minimizing the cross-entropy error between the predicted distribution at each node and the target distribution at the same node.

## 5.5. Software Development

BEIA developed the following software programs in C#,

* 1. SoMeDi\_Sentiment-Analyze\_MS-Azure\_EN - using MS-Azure Services, in English
  2. SoMeDi\_Sentiment-Analyze\_MS-Azure\_RO - using MS-Azure Services, in Romanian
  3. **SoMeDi\_Sentiment-Analyze\_StanfordCoreNLP\_EN - using StanfordCoreNLP Tools, in English,**
  4. SoMeDi\_Sentiment-Analyze\_StanfordCoreNLP\_RO - using StanfordCoreNLP Tools, in Romanian.

The programs are presented as Installation programs for Windows 10, x64.

Folder: “Doc. Install-3 programs.zip” contains these installation programs.

Notes:

* No shortcuts for programs
* The folder “ stanford-corenlp-3.9.1-models” is necessary to be placed on : C:\Program Files\BEIA\SoMeDi\_Sentiment-Analyze\_StanfordCoreNLP\_EN, after installation.
* MS\_Azure services need a MS acount; the programs do not work without this (remote server returns error).

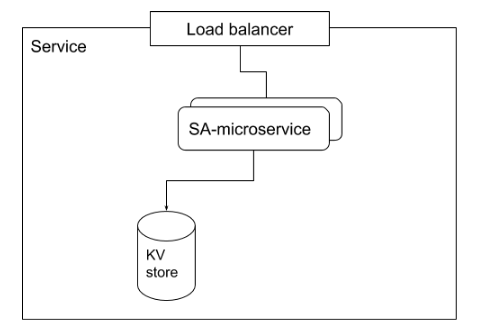
## 5.6. Integration with the SoMeDi platform

In this section, we present the methodology to integrate the DII tool as a web application in order to ensure better usability of the SoMeDi platform.

### Overview

The sentiment analysis is designed as a microservice, in order to ensure the scalability requirements, but also allow modularity and reusability,

The microservice architecture is presented in the diagram shown below, and described as follows:



1. A request comes from a load balancer, provided by docker.
2. The request reaches one of the containers and starts being processed.
3. A job is stored in a key-value store. The job will also contain the result, as returned by the SA engine.
4. The result is returned to the client, either in the same request or later as the status of a job.
5. Jobs expire after a specified timeout and are removed from the key-value store.

The key-value store engine used is Redis. It is used because it can allow persistence and if needed it can be replicated on multiple nodes.

The microservice is implemented as a node.js application. The service is modular and can support any of several sentiment analysis engines. The interaction to each engine is implemented as a class: GoogleSA.js, AzureSA.js, StanfordSA.js.

### API contract

The microservice can be consumed as a JSON REST API.

**Endpoint**: /health-check

**HTTP Method**: GET

**Scope**: internal

**Remarks**: Used by the load-balancer to decide which instances of the microservice are ready to serve traffic. It is not exposed externally

**Endpoint**: /sentiment-analysis

**HTTP Method**: POST

**Scope**: public

**Request Body:**

{

“token”: <authentication token>,

“engine-hint”: “google”|”azure”|”stanford”

“language”: <language>|”auto”,

“content”: <content for sentiment analysis>

}

**Response Body:**

**{**

“status”: “success”|”invalid-arguments”|”server-error”|”pending”|”rate-limit”|”access-denied”,

“job-id”: <UUID of job>,

“sentiment-score”: <floating point number representing score>

}

**Endpoint:** /job-status

**HTTP Method:** POST

**Scope**: public

**Request Body:**

{

“token”: <authentication token>,

“job-id”: <UUID of job>

}

**Response Body:**

{

“status”: “success”|”invalid-arguments”|”server-error”|”pending”|”rate-limit”|”access-denied”,

“job-id”: <UUID of job>,

“sentiment-score”: <floating point number representing score>

}

Each endpoint is guaranteed to always reply within 50 ms. The reply is either a success message containing the sentiment score or another message containing a job id that can be later checked for status. Each endpoint accepts a maximum size of 2 kB of JSON payload. Invalid or unknown fields are ignored.

In case multiple engines are enabled in the service “engine-hint” can allow selecting a specific one. If there is only one engine used by the backend or otherwise available, this parameter is ignored. In general, this parameter can only be treated as a hint. The service doesn’t guarantee a specific engine will actually be used.

To prevent abuse of the service, each token will have associated a limited number of requests per minute, with a possibility for burst in the first minute. For example, a token might have a limit of 100 requests/minute and a burst of 300 requests in the first minute, after a period of no traffic for that token. These limits will be imposed at service level and in case a limit is exceeded, a “rate-limit” status will be returned as a response. The theoretical model is that of a token bucket.

### Security

The service communicates through TLS v1.2. Because the microservice can potentially be hosted in a public cloud, and the data might to traverse the public internet, an authentication token must used. For the current implementation the token is statically generated by the admin. More advanced use cases can be devised later.

### Compliance

The service is stateless except for the sentiment score and job id which might be stored locally for a limited time. No other data, including user identifiable data will be stored by the service.

### Telemetry

The service generates metrics which can be used to assess the health of the service. All the following values have another dimension on which they can be split which is the microservice instance identifier, in this case the host allocated by docker to the container instance.

|  |  |
| --- | --- |
| **Metric name** | **Value meaning** |
| sa-service.loop-delay | The loop delay of the node.js event loop. A high value represents high load. |
| sa-service.<engine>-request-time | The time it takes the engine to compute the sentiment score. It is a statistic type and it will contain p50, p90, p99, mean, min, max sub-values. Engine variable is replaced by the actual engine used: Google, Azure, Stanford. |
| sa-service.<endpoint>.<status>.count | A counter per interval for each of status messages returned, for each of the endpoints. |
| sa-service.content-size | The content-size sent to the service. It is a statistic type and it will contain p50, p90, p99, mean, min, max sub-values. |
| sa-service.memory-used | The memory used by the service. |

### Sentiment Analysis Service Communication

As mentioned in the previous section, the SA microservice is implemented as a node.js application and can be consumed as a JSON REST API, having the following web address:

*<https://somedi-api.beia-consult.ro> .*

To call the NLP service, the following features (functions) have been implemented in the OctoberCMS platform :

The integration of the sentiment analysis microservice with the SoMeDi Recruiting platform required the following implementation of the following features (functions implemented in the October CMS platform):

* 1. createNlpJobId (which returns the job id after the security token and content has been sent)

public function createNlpJobId($nlpQuestion = null)

{

$nlpURL = 'https://somedi-api.beia-consult.ro/';

$callData = 'sentiment-analysis';

$nlpToken = '4kNjyhGmhg1aii6XNJnbgn2auFrIQvTn';

$nlpLanguage = 'auto';

$nlpEngine = 'google';

$data = array("token" => $nlpToken, "engine-hint" => $nlpEngine, "language" => $nlpLanguage, "content" => $nlpQuestion);

$data\_string = json\_encode($data);

$ch = curl\_init($nlpURL . $callData);

sleep(2);

curl\_setopt($ch, CURLOPT\_POST, true);

curl\_setopt($ch, CURLOPT\_HTTPHEADER, array('Content-Type', 'application/json'));

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $data\_string);

curl\_setopt($ch, CURLOPT\_CONNECTTIMEOUT, 15);

if (null !== $this->timeout) {

curl\_setopt($ch, CURLOPT\_TIMEOUT, $this->timeout);

}

if ($this->proxy) {

curl\_setopt($ch, CURLOPT\_HTTPPROXYTUNNEL, true);

curl\_setopt($ch, CURLOPT\_PROXY, $this->proxy);

}

$response = curl\_exec($ch);

curl\_close($ch);

$response = json\_decode($response,true);

return $response;

}

* 1. getNlpScore (returns the Sentiment Analysis score based on the recieved job id)

$nlpURL = 'https://somedi-api.beia-consult.ro/';

$callData = 'job-status';

$nlpToken = '4kNjyhGmhg1aii6XNJnbgn2auFrIQvTn';

$data = array("token" => $nlpToken, "job-id" => $nlpJobId);

$data\_string = json\_encode($data);

$ch = curl\_init($nlpURL . $callData);

sleep(2);

curl\_setopt($ch, CURLOPT\_POST, true);

curl\_setopt($ch, CURLOPT\_HTTPHEADER, array('Content-Type', 'application/json'));

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $data\_string);

curl\_setopt($ch, CURLOPT\_CONNECTTIMEOUT, 15);

if (null !== $this->timeout) {

curl\_setopt($ch, CURLOPT\_TIMEOUT, $this->timeout);

}

if ($this->proxy) {

curl\_setopt($ch, CURLOPT\_HTTPPROXYTUNNEL, true);

curl\_setopt($ch, CURLOPT\_PROXY, $this->proxy);

}

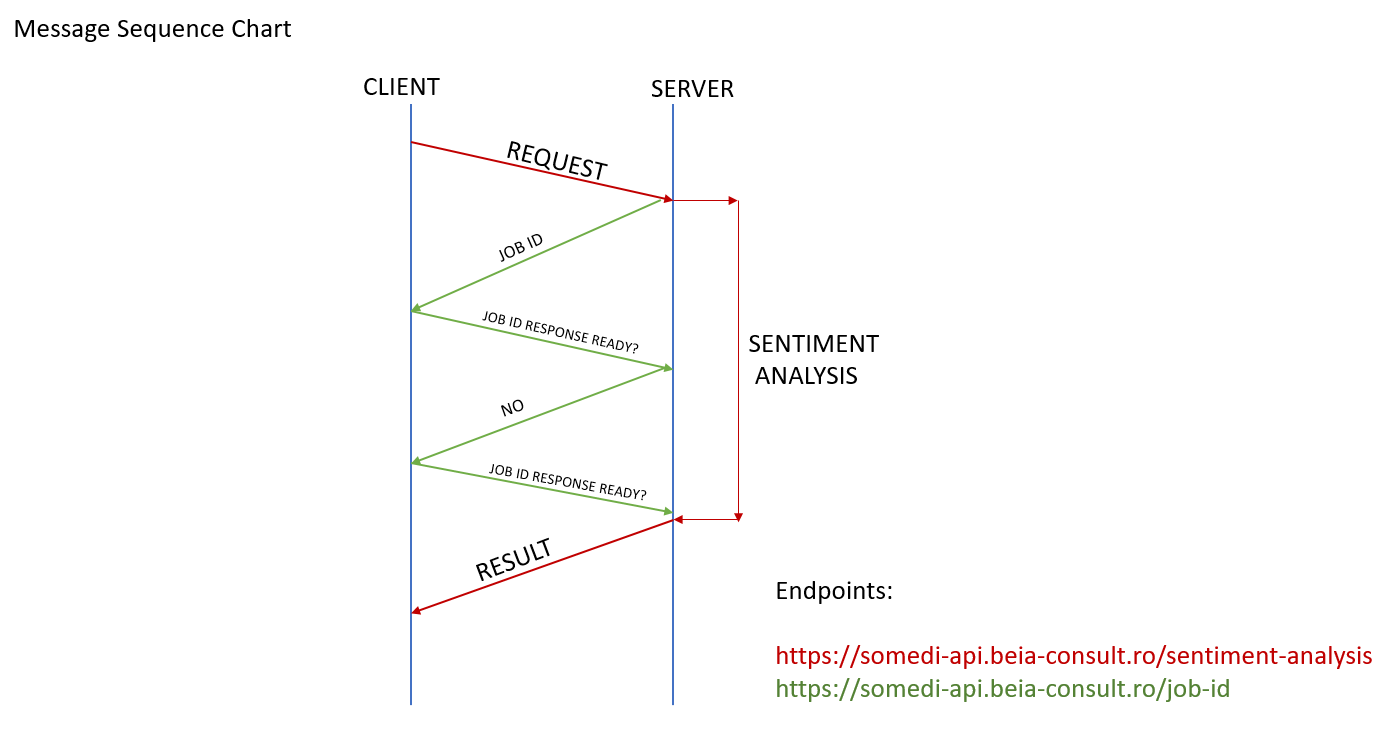
$response = curl\_exec($ch);

curl\_close($ch);

$response = json\_decode($response,true);

return $response;

}



Schematics of the message sequence for the SA microservice (Recruiting use case)

**Chapter 5 described the main achievements with regard of Deliverable D3.2, presenting the first version of the DII toolkit specific to the Romanian Use Case - The Stanford NLP project for EN version is available here[[32]](#footnote-77).**

**The testing and validation processes of the DII tool advanced in this phase are detailed in Deliverable D4.3.**

# 6. Aligning metadata intelligence with recruitment use case

This chapter will address the main tasks specific to **Deliverable D3.3 DII Metadata intelligence and Model based techniques**. These tasks precede the Demonstrators release in WP4.

In order to prepare the Recruiting Demonstrator release in WP4, the above-described NLP applications will be tested on as many candidates as possible, and so Digital Interaction Data will be created.

These DID will be structured as Metadata (DataBase) and then processed using Data Mining type Clustering and Text Analytics methods to find the following information / patterns:

1. identify the most suitable method for finding the candidates’ opinions about the hiring company fields of activity (a comparison between the three NLP solutions Stanford, Google, Azure);
2. produce several visual instruments (reporting tools) with statistics concerning:
   1. the internship programme – candidates age, field of study, level of study, work experience;
   2. the candidates’ opinions about the hiring company fields of activitiy;
   3. the number of accepted applications reported to the number of candidated who actually started the internship programme;
   4. the candidates’ opinions after the internship programme (feedback).

# 7. Aligning metadata intelligence with marketing use case

[Hi Iberia & TAIGER]

# 8. Aligning metadata intelligence with NBA use case

[Turkcell and EVAM]

# 8.Conclusions

# 

# Bibliography

Liu, B., 2012. *Sentiment analysis and opinion mining.* s.l.:Morgan & Claypool publishers.

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

Graves, A., 2013. Generating sequences with recurrent neural networks. *arXiv:1308.0850.*

Karpathy, A., Johnson, J. & Fei-Fei, L., 2015. Visualizing and understanding recurrent networks. *arXiv:1506.02078.*

Kim, Y., Jernite, Y., Sontag, D. & Rush, A. M., 2015. Character-aware neural language models. *arXiv:1508.06615.*

Radford, A., Jozefowicz, R. & Sutskever, I., 2017. Learning to generate reviews and discovering sentiment. *arXiv:1704.01444.*

Sutskever, I., Vinyals, O. & Le, Q., 2014. Sequence to sequence learning with neural netowrks. *Advances in neural information processing systems,* pp. 3104-3112.

Yang, Z., Salakhutdinov, R. & Cohen, W. W., 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. *arXiv:1703.06345.*

1. "ILSVRC2015 Results - ImageNet." <http://image-net.org/challenges/LSVRC/2015/results>. Accessed 2 Jun. 2017. [↑](#footnote-ref-46)
2. "ImageNet - Wikipedia." <https://en.wikipedia.org/wiki/ImageNet>. Accessed 2 Jun. 2017. [↑](#footnote-ref-47)
3. "VisualGenome." <http://visualgenome.org/>. Accessed 2 Jun. 2017. [↑](#footnote-ref-48)
4. "MS-Coco." <http://mscoco.org/>. Accessed 2 Jun. 2017. [↑](#footnote-ref-49)
5. "[1512.03385] Deep Residual Learning for Image ... - at www.arxiv.org.." 10 Dec. 2015, <https://arxiv.org/abs/1512.03385>. Accessed 2 Jun. 2017. [↑](#footnote-ref-50)
6. "DenseCap: Fully Convolutional Localization Networks for Dense ...." 24 Nov. 2015, <https://arxiv.org/abs/1511.07571>. Accessed 2 Jun. 2017. [↑](#footnote-ref-51)
7. "VisualGenome." <http://visualgenome.org/>. Accessed 2 Jun. 2017. [↑](#footnote-ref-52)
8. "Chapter 9: Convolutional Networks - Commonlounge." <https://www.commonlounge.com/discussion/3040dd68b6454695a63cbfbb78d2e557/post>. Accessed 2 Jun. 2017. [↑](#footnote-ref-53)
9. "Deep Learning Book." <http://www.deeplearningbook.org/>. Accessed 2 Jun. 2017. [↑](#footnote-ref-54)
10. <http://torch.ch/> - Torch Deep Learning Framework website. [↑](#footnote-ref-55)
11. <https://pytorch.org/> - PyTorch, Python front-end for the Torch framework. [↑](#footnote-ref-56)
12. <https://nodejs.org/en/> - Node.js server side framework. [↑](#footnote-ref-57)
13. https://conda.io/docs/ [↑](#footnote-ref-58)
14. https://pytorch.org/ [↑](#footnote-ref-59)
15. https://github.com/allenai/allennlp [↑](#footnote-ref-60)
16. https://github.com/zalandoresearch/flair [↑](#footnote-ref-61)
17. https://github.com/fastai [↑](#footnote-ref-62)
18. https://github.com/google-research/bert [↑](#footnote-ref-63)
19. https://www.clips.uantwerpen.be/conll2002/ner/ [↑](#footnote-ref-64)
20. http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/ [↑](#footnote-ref-65)
21. http://www.sepln.org/workshops/tass/2012/about.php [↑](#footnote-ref-66)
22. http://www.sepln.org/workshops/tass/2018/ [↑](#footnote-ref-67)
23. http://flask.pocoo.org/ [↑](#footnote-ref-68)
24. https://www.docker.com/ [↑](#footnote-ref-69)
25. https://en.wikipedia.org/wiki/Docker\_(software) [↑](#footnote-ref-70)
26. <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html> [↑](#footnote-ref-71)
27. <https://web.stanford.edu/~jurafsky/slp3/4.pdf> [↑](#footnote-ref-72)
28. <https://web.stanford.edu/~jurafsky/slp3/10.pdf> [↑](#footnote-ref-73)
29. <https://www.nltk.org/book/ch05.html> [↑](#footnote-ref-74)
30. <http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/> [↑](#footnote-ref-75)
31. <https://medium.com/@josecamachocollados/on-the-contribution-of-neural-networks-and-word-embeddings-in-natural-language-processing-c8bb1b85c61c> [↑](#footnote-ref-76)
32. <https://drive.google.com/drive/folders/1lcjed8La3QqT7eF4hAf9P1T2PnLhCeLv?usp=sharing> [↑](#footnote-ref-77)