UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

# título del TFG
# Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — May 13, 2019
Tutor: nombre tutor

# Contents

# List of Figures

# List of Tables

*Apéndice A*

# Software plan

## A.1 Introduction

In the following annex, the organizational aspects of the study of different NER classifiers and the development of the software are documented. More precisely, the software development process and tools that were used to manage the process are described, followed by a examination of the course of the project. The second part of the annex examines the project's viability, including the calculation of involved costs and profit possibilities.

## A.2 Project Management

**Scrum** The project's management is inspired by the Scrum model used in agile software development. The model is based on the assumption that projects are too big to be planned in it's entirety at the start. Therefore only a rough outline is made at the start. The project is divided into several milestones that provide an agile approach.

Scrum is a team based approach to project management. Due to the fact that this bachelor thesis is only written by one person the majority of the concepts can't be applied exactly as intended by Scrum. Consequently the project management approach is only loosely based on Scrum.

One concept that is applied are Sprints. In this case most sprints cycles had a duration of approximately a month. Some are bigger and some are smaller due to the complexity of tasks at hand and the time available. Sprint meetings between the author and the project's coordinator were held every two weeks, usually around the middle and end of each sprint. In the meeting in the middle the tasks progress was discussed, while in the meeting at the end the results of the sprint was discussed and the next sprint was vaguely planned. The second meeting can therefore be seen as the Sprint Planning and Sprint Review. The project's coordinator can be seen as the Project Owner of the Scrum model, prioritizing tasks and guiding the project's direction.

**Github and ZenHub** The project is hosted on github. To organize tasks ZenHub was used. Zenhub provides a board to visualize tasks as well as it provides an overview of the remaining workload. It also offers several tools to inspect work velocity. Each Github issue is assigned a amount of story points, estimating the tasks size. In this case each point is the equivalent of the workload of <points hours.

## A.3 Time plan

The Kick-Off Meeting took place in the second week of December 2018. The elemental ideas of the project were discussed. Due to exams and other private responsibilities the project wasn't directly started after the meeting. Instead the 9. of January marked the beginning of the project.

Some milestones were smaller than others in a similar time frame. This is due to responsibilities of other classes and exam periods which reduced time availability during the semester.
The next paragraphs give an overview over the the phases of development.

### Milestone 1 (9.01-14.01.2019): Initial project setup/ Research regarding NLP

In the first relatively small Milestone the projects infrastructure was set up. Also some research regarding the theoretical concepts of NLP and NER were done and documented.

### Milestone 2 (8.02-31.02.2019): NLTK-Experiments/ Preprosessing Data

The second Milestone consisted of researching and experimenting with one of the libraries (NLTK) used later on in the project. An initial NER chunker was introduced and functions prepossessing the Data were written. Ideally this milestone would have been bigger and the project would have advanced much more, but my laptop broke and had to be send in. Due too the bad infrastructure of the university not much could have been worked on.

### Milestone 3 (01.03-08.04.2019): NLTK

Milestone 3 was the biggest Milstone to that point. Things got serious as different NLTK chunkers got introduced and a training script was wtitten. Inicially the due date was the end of march. But some complications in having to figure out how to train NER chunkers without any use cases in the internet and the midterm exam period it had to be postponed by a week.

### Milestone 4 (09.04-31.04.2019): Other Classifiers

In the inicial plan it was planned to create a few other classifiers with other libraries such as Stanford Core, Sklearn and OpenNLP. But that plan changed a bit. Together with the projects coordinatior the decision to develop a different classifier using more advanced machine learning techniques was made. Do to the postponement of the last milesone, exams and working on a bigger project in another class most of april couldn't be used for this milestone as the commit history suggests. The project of the other class was related to this theme as it covered sentiment analysis with the sklearn library. The concepts learned and applied there were adapted towards named entity recognition and incorporated into this project.

### Milestone 4 (01.05-30.05.2019): Evaluation

### Milestone 5 (01.06-31.06.2019): Endstage

## A.4 Feasibility study

### Economic viability

### Legal Feasibility

*Apéndice B*

# Requirements Specification

## B.1 Introduction

This chapter contains requirement specifications for the developed software. It provides an overview over the software's required functionality.

## B.2 General Objectives

The general objective of the software is to find named entities in noisy texts. Different classifiers ...

## B.3 Requirements Catalogue

This section lists the software's functional requirements.

**Actors** The only actor of the software is it's user. There are no different roles an actor can have, as the only actor he has the whole feature-set of the program available. His goal is to get as accurate named entity predictions as possible.

## B.4 Requirements specification

Table B.1: FR «Process input»

| **ID:** | **Name: Process input** |
|---|---|
| Description | The system reads Text that is inputted by the user and processes it into the data formats processable by the different classifiers |
| Process description | Whenever new Text is entered it is read and converted into a format readable by the system in order to work with that data. The data formats required are described in section x. |
| Main path (M) | 1. User selects . . . <br> 2. System demands . . . |

Table B.2: FR «Output results»

| **ID:** | **Name: Output results** |
|---|---|
| Description | After classification the system outputs it's classification results. |
| Process description | After classification the results are outputted in two different ways. |
| Output Formats | 1. **Json File:** The results are saved in a Json file. Each row contains one word. An empty line represents the end of a sentence. After the word follows the predicted class. Word and class are separated by a space. <br> 2. **Image** For better visualization the results are also outputted as an image. The image contains the original text with found entities being highlighted. |

Table B.3: FR «Classification»

| **ID:** | **Name: Classification** |
|---|---|
| Description | The selected classifier predicts the named entities inside a text. |

Table B.4: FR «Selection of Classifier»

| **ID:** | **Name: Selection of Classifier** |
|---|---|
| Description | A Classifier can be selected out of a predefined selection of different classifiers. |
| Options | 1. option1<br>2. option2<br>3. optionN |

Table B.5: FR «Fetch random tweet»

| **ID:** | **Name: Fetch random tweet** |
|---|---|
| Description | A random tweet is fetched to highlight classification |
| Process Description | x |

*Apéndice C*

# Design specification

## C.1 Introduction

## C.2 Data design

## C.3 Precedural design

## C.4 Architectural design

*Apéndice* $D$

# Technical Programming Documentation

## D.1 Introduction

## D.2 Directory structure

## D.3 Programmer's Manual

## D.4 Compilation, installation and execution of the project

## D.5 System tests

# *Apéndice E*

# User documentation

## E.1 Introduction

## E.2 User requirements

## E.3 Installation

## E.4 User manual

## E.5 Data

This section describes how the Data is saved and preprocessed. The description is divided into two sections because the NLTK classifiers and the pytorch classifier read and process Data in a different way.

### Data description

The following descriptions only contain a description of the input formats of the different datasets.

#### Dataset Formats

**WNUT 2017 Format** Each line of original Data describes one token (word). Each line contains the word and it's entity.
Example:
@Suzie55 O

whispering O
cause O
I O
may O
have O
had O
1 O
too O
many O
vodka B-product
’s O
last O
night O

### NLTK

**Input Format** The input format of the file for the NLTK classifiers contain a token per line. A line is made up by the word, it’s POS tag and named entity tag with IOB tagging scheme.
Example:
@Suzie55 JJ O
whispering NN O
cause NN O
I PRP O
may MD O
have VB O
had VBD O
1 CD O
too RB O
many JJ O
vodka NN B-product
’s POS O
last JJ O
night NN O

**Preprocessing** The following steps are necessary to transform the data into the NLTK classifiers input format:

- Remove entity Tags: All entity tags have to be remove in order to tokenize file later on
- Tokenization: Tokenize Datasets into sentences and then words
- POS Tagging: POS tag resulting words from previous step
- Join pos tagged results with previously removed entity tags

## Pytorch

**Input Format** The dataset is divided into two different text files. One containing all it's sentences (sentences.txt), each line containing one sentence, the other containing their associated labels.
Example (Representation of a sentence):
sentences.txt: @Suzie55 whispering cause I may have had 1 too many vodka 's last night and am a lil fragile , hold me ?

labels.txt: O O O O O O O O O O B-product O O O O O O O O O O O O

**Output Format**

The outout format of all classifiers look the same for easy use of results. Each line describes one token, containing it's word and entity in IOB tagging scheme. Sentences are devided by an empty line.
Example: @Suzie55 O
whispering O
cause O
I O
may O
have O
had O
1 O
too O
many O
vodka B-product
's O
last O
night O