MASK Framework

Nikola Milosevic

Jun 04, 2019

CONTENTS:

CHAPTER

ONE

INTORDUCTION

MASK Framework is an open-source framework for de-identification of medical free-text data

In this project, we will develop an open-source framework for automated de-identification of medical textual data. Such data contains information that can be utilized to support clinical research, but its native form contains sensitive personal identifiable information (PII) that should not be accessed by anyone who does not provide direct clinical care.

The project aims to enhance the current processes and build an open-source platform that can be used for flexible masking of personal information, ensuring that de-identified medical text still contains enough information to facilitate research.

In order to facilitate flexibility, the de-identification system has to be configurable by the user in terms of:

- Types of PII that have to be identified in free-text data;
- Approaches to masking of the identified data (keep, redact, map, etc.);
- Disclosure risk analysis that is performed on the data;
- The methodology that is applied for each of the steps.

CHAPTER

TWO

CLASSES AND FUNCTIONS

mask_framework.py – Main MASK Framework module

class mask_framework.**Configuration**(*configuration='configuration.cnf'*)
Class for reading configuration file

Init function that can take configuration file, or it uses default location: configuration.cnf file in folder where mask_framework is

mask_framework.**main**()
Main MASK Framework function

ner_plugins - a set of modules that can perform named entity recognition. Basically, plugins for different kinds of named entity recognition

class mask_framework.**Configuration**(*configuration='configuration.cnf'*)
Class for reading configuration file

Init function that can take configuration file, or it uses default location: configuration.cnf file in folder where mask_framework is

class ner_plugins.NER_CRF.**NER_CRF**
The class for executing CRF labelling based on i2b2 dataset (2014).

custom_span_tokenize(*text*, *language='english'*, *preserve_line=True*)
Returns a spans of tokens in text.

Parameters

- **text** – text to split into words
- **language** (*str*) – the model name in the Punkt corpus
- **preserve_line** – An option to keep the preserve the sentence and not sentence tokenize it.

custom_word_tokenize(*text*, *language='english'*, *preserve_line=True*)
Return a tokenized copy of *text*, using NLTK's recommended word tokenizer (currently an improved TreebankWordTokenizer along with PunktSentenceTokenizer for the specified language).

Parameters

- **text** – text to split into words
- **text** – str
- **language** (*str*) – the model name in the Punkt corpus
- **preserve_line** – An option to keep the preserve the sentence and not sentence tokenize it.

doc2features(*sent*)
Transforms a sentence to a sequence of features

Parameters **sent** – a set of tokens that will be transformed to features

perform_NER(*text*)
Implemented function that performs named entity recognition using CRF. Returns a sequence of tuples (token,label).

Parameters **text** – text over which should be performed named entity recognition

tokenize_fa(*documents*)
Tokenization function. Returns list of sequences

Parameters **documents** – list of texts

word2features(*sent*, *i*)
Transforms words into features that are fed into CRF model

Parameters

- **sent** – a list of tokens in a single sentence
- **i** (*int*) – position of a transformed word in a given sentence (token sequence)

class ner_plugins.NER_abstract.**NER_abstract**
Abstract class that other NER plugins should implement

perform_NER(*text*)
Implementation of the method that should perform named entity recognition

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

n