Reducing supervision in Sequence Labeling

Master Dissertation

For the degree of MSc in Computer Science

School of Computer Science, University of Birmingham

UNIVERSITY OF
BIRMINGHAM

Supervisor: Dr Giorgio Orsi

Author: Juan Carlos Jiménez Illescas

Student ID 1536684

September 12, 2016
Index of Contents

Abstract ................................................................................................................................. 5
Acknowledgements ................................................................................................................ 6
Introduction .......................................................................................................................... 7
Chapter 1. Background knowledge .................................................................................... 9
  Foundations ......................................................................................................................... 9
  Related work ....................................................................................................................... 9
    Aggregating Semantic Annotators (Chen, Ortona, Orsi, & Benedikt) ......................... 9
    Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random
    Fields (Mann & McCallum, 2008) ................................................................................. 9
    Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence
    Data (Lafferty, McCallum, & Pereira, 2001) ................................................................. 10
    Shallow Parsing with Conditional Random Fields (Sha & Pereira, 2003) ................. 10
Chapter 2. Problem analysis and specification .................................................................. 11
  Problem definition ............................................................................................................. 11
  Problem approach ............................................................................................................ 11
    System requirements (Sommerville, 2011) (Bahsoon, 2015) .................................... 11
Chapter 3. System Design .................................................................................................. 13
  Components Interaction .................................................................................................. 13
  Handling the user interaction with the system (Microsoft; Microsoft) ......................... 14
  Package definition ........................................................................................................... 15
  User interface Design ...................................................................................................... 15
Chapter 4. Implementation and Testing ............................................................................. 17
  Rationale of the APIs selection ....................................................................................... 17
    Election of ROSeAnn as NER provider ...................................................................... 17
    Election of CRF Project from sourceforge as sequence labeling tool ...................... 18
  Usage of selected APIs .................................................................................................... 19
    Use of ROSeAnn API ................................................................................................. 19
    Use of CRF .................................................................................................................. 19
  Implementation of the feedback loop .............................................................................. 19
  Rationale of the Data structures selection ................................................................... 20
    Data Structures related to CRF API .......................................................................... 20
    Data Structures related to the feedback section ......................................................... 20
  Integration approach in the project .............................................................................. 20
  Feedback approach in the project ............................................................................... 21
Graphic User Interface Implementation .................................................. 21
Testing strategy .......................................................................................... 22
White box testing ....................................................................................... 22
Black box testing ....................................................................................... 22

Chapter 5. Project Management ................................................................ 23

Chapter 6. Results and evaluation .............................................................. 24
Results of testing ....................................................................................... 24
White box tests results .............................................................................. 24
Black box tests results .............................................................................. 28
Evaluation .................................................................................................. 29

Chapter 7. Discussion and conclusions ..................................................... 30
Discussion .................................................................................................. 30
Project Summary ....................................................................................... 30
Possible improvements ............................................................................. 30
Potential users of the system .................................................................... 30
Conclusions ............................................................................................... 30

References and Bibliography ..................................................................... 31

Appendix A. User's Guide .......................................................................... 32
Training the system .................................................................................... 33
Testing the system ..................................................................................... 34
Provide feedback to the system ................................................................. 34

Appendix B. How to run the code .............................................................. 36

Appendix C. SVN content explanation ....................................................... 37

Index of Figures

Figure 1 Stanford Named Entity Tagger .................................................... 7
Figure 2 Black box model of the system .................................................... 13
Figure 3 System decomposition ................................................................. 14
Figure 4 Model-View-Controller (Microsoft; Microsoft) ................................ 14
Figure 5 Class Diagram ............................................................................. 15
Figure 6 Graphic User Interface Design .................................................... 16
Figure 7 Detailed components Diagram of the system ............................... 17
Figure 8 Linear Chain Comparison ............................................................. 18
Figure 9 Graphic User Interface Implementation ....................................... 21
Figure 10 Code to measure the performance of the training phase ............. 24
Figure 11 Code to measure the performance of the testing phase ............... 24
Figure 12 Training phase results .......................................................... 25
Figure 13 Testing phase results ............................................................ 26
Figure 14 Code to display the training artifact ....................................... 26
Figure 15 Training artifact content ......................................................... 26
Figure 16 Code to test the feedback segment .......................................... 27
Figure 17 Feedback example .................................................................. 27
Figure 18 Entry created by feedback segment ......................................... 27

Index of Tables

Table 1 Testing Plan ............................................................................. 22
Table 2 Tests Results Summary ............................................................ 28
Abstract

Nowadays information extraction is an important application in Artificial Intelligence topic. This task is especially complex considering that lexical recognition stills one of the most complex process of the human being. In this project, we learned the aspects related to a particular categorization of words known as Named Entity Recognition, whose aim is to group entities according to the function embedded in it. We also learned about probabilistic algorithms, called Conditional Random Fields, used to perform labeling tasks on different datasets.

We conducted white-box testing as well as black-box testing on the software implemented to assess the final product and determine the usability of the tool. The final version of the product includes a GUI developed in Java, and it achieves a reduction of the supervision in the training phases and in the verification phase of the labeling process.
Acknowledgements

I would like to thank to God because his mercy has brought me here.

I would like to thank my mother and my grandparents for not giving up on me.

I would like to thank my blackbird for standing by me. Your love and friendship make me a better person allowing me to achieve bigger things every day.

I would like to thank University of Birmingham because of the excellent academic training that this institution has given me.

I would like to thank my sponsor, the Secretaria de Educación Superior, Ciencia, Tecnología e Innovación (Secretary of Higher Education, Science, Technology and Innovation) for giving the Ecuadorian people the opportunity to study in the most prestigious Universities of the world.
Introduction

The information extraction process is an important part of data processing and an application of Artificial Intelligence that is becoming increasingly important. Although verbal communication is natural for humans, the task of text recognition is very complex to be modelled and approached for computer systems.

Some major advances have been made in this task, the conception and the availability of NERs (Named Entity Recognizers) as APIs that could be consumed through Internet gives to the developers the opportunities to design more complex programs regardless the implementation used for NER, obtaining systems that could categorize entities found in a piece of text separating them into different groups.

As example of this, the following figure shows the web interface for the Stanford Named Entity Recognizer in action (Stanford), considering a piece of text.

![Stanford Named Entity Tagger](image)

**Figure 1 Stanford Named Entity Tagger**

Although the accuracy in the results is improved according to new implementations and strategies to approach the problem are developed, there are some levels under which some incorrect labels could be obtained, and in this case. The approach to correct these mistakes is that a person (user) should manually modify the label assigned to an entity, strategy that relies on some subjective, and the fallible, parameters e.g. the knowledge and criteria of the user.

The aim of this project is to develop a system which limits the feedback of the user to minimize possible incorrect categorization of entities, using sequence labeling tools that traditionally used with labelled values as trainers. In this project we also will show how we can integrate available solutions to avoid the manual labeling of the training datasets reducing the supervision as well.

The following sections of the dissertation include the information as described in the following text:

1. Chapter 1. Background knowledge expanding the basic foundations related to Artificial Intelligence and NERs, also provides small summaries of related papers.
2. Chapter 2. Problem definition and requirements specification.
3. Chapter 3. Design process describing the components interaction, class diagrams, the MVC pattern used to handle the user interactions and the design of the GUI of the system.

4. Chapter 4. Implementation description. Rationales about the selections of APIs, the way that the components interact among them, how the feedback is achieved, the GUI implementation description and the testing strategy.

5. Chapter 5. The approach to manage the project, a brief description of the tasks performed to implement the system and the model used, since a software engineering perspective.

6. Chapter 6. Contains the tests results and evaluation of the final product obtained.

7. Chapter 7. Contains a discussion summarizing the project achievements, identifying possible improvements in the software, description of the system’s usage and finally the conclusions derived from the realization of the project, including the coding tasks and the literature review.

All software for this project can be found at https://codex.cs.bham.ac.uk/svn/projects/2015/jcj584/.
Chapter 1. Background knowledge

Foundations

In the actual world the availability of big volume of information requires that the information processing to be more independent of the supervision of the human being. In this sense, machine learning offers a wide perspective to handle large amount of data that in the past could not been automated at the same degree, basically due to the limitations in hardware components in computational systems. Formally defined by Arthur Samuel (1959) as the “Field of study that gives computers the ability to learn without being explicitly programmed”, nowadays considering the actual scale of integration in the processors and memory, this field has experienced an increase in researches and applications.

As an important part of the machine learning paradigm, the multiclass classification allows to the system to discriminate collections of objects upon some features or characteristics separating them into more than two classes. As stated by (Rifkin, 2008), upon a given set of given examples in the training set, “The goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs.”

In this project, the multiclass classification task will be focused into sequence labeling where an arbitrary sequence of text is given, the system will classify each one of the components, called entities, of this text considering defined categories (Sang, 2002). Is important to notice that the categories are not defined by the system itself, instead it takes the classes as defined in web tools known as Named Entity Recognizers (NERs).

Related work

Some significant contributions related to the automation in labeling process have been done by various researches; those that are more closely related to the aim of the project are the following papers.

Aggregating Semantic Annotators (Chen, Ortona, Orsi, & Benedikt)
The aim of this paper is to propose, implement and assess an algorithm that reconciles the various annotations of a given span, i.e. given a span where different online NERs could define with diverse annotations, the algorithm should determine what the final value of the annotation should be, considering accuracy and the semantic content of the annotations as well as the span.

The implementation of this algorithm led to the realization of the software ROSeAnn, which is available and documented. (DIADEM)

The paper proposes a semi-supervised learning approach to perform the sequence labeling task using conditional random fields as the mathematical tool to model the system behavior. By using this perspective, the modeling of the system is done considering features instead of instances. The implementation of this algorithm is available online and is part of the Machine Learning API (Application Programming Interface) called Mallet. (UMASS)

This paper proposes a perspective based on Conditional Random Fields (CRF) to label and segment data. The authors in the abstract state “Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graphical models, which can be biased towards states with few successor states.”

Over these considerations, the CRF implementation algorithms are preferable to perform labeling tasks.

Shallow Parsing with Conditional Random Fields (Sha & Pereira, 2003)

The content of this paper describes the implementation of the CRF in order to obtain some measurable results to be compared with traditional implementations such as HMM. This paper constitutes the base upon which the CRF project package available at Sourceforge page, containing the documentation (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Project Page) and could be also downloaded (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Sourceforge).
Chapter 2. Problem analysis and specification

Based on the feedback type given to the system, this could be classified into four main categories, which are described and discussed in (Russell & Norvig, 2010) and are:

1. Unsupervised Learning. No feedback is explicitly given to the system.
2. Reinforcement Learning. The system receives a series of reinforcements, either reward or punishment.
3. Supervised Learning. Given an example of inputs and outputs, the system approximates a function that models their behavior and could apply to a new set of inputs to determine their output.
4. Semi-supervised learning. The most frequent in practice, we have data examples, but not the whole examples are labeled or contain some errors. This type of error is called noise in the samples.

Problem definition

The actual state of the labeling systems, under supervised conditions, requires that the trainer gives to the system a sequence of labeled inputs and the corresponding outputs. The next stage corresponds to model the system, and then apply the model in a new data set called test set. The final phase of the process consists in a human being manually reclassifying each entry, labeling by hand the entities.

The main limitations in this perspective are detailed as follows:

1. The trainer must ensure that the training set fits the required format for the system, which translates into time consuming activity.
2. The training set could be corrupted and then the model that the system learns will not correspond to the appropriated relation between inputs and outputs, this could introduce significant bias in the results.
3. The manual reclassification could also introduce incorrect results, because the spans could led to ambiguous classes or even groups that are not defined in NERs, e.g. The span “Barack Obama”, since the NER’s perspective corresponds to the category PERSON, however the reviewer could categorize it as PRESIDENT, however this category could not be defined as a group. Another issue that could arise in this reclassification process is that a more specific group could exist to classify an entity but the user selects a more general one, e.g. considering the entity “Barack Obama”, let’s suppose there is a new category called PRESIDENT, then the entity should be into this class, however the reviewer considers that the category PERSON is more appropriated or simply ignores the fact that the PRESIDENT category exists.

Problem approach

To diminish the possible negative impacts that the limitations could have in the results of the labeling process, it is important to define certain strategies to minimize the probability of errors in the results.

System requirements (Sommerville, 2011) (Bahsoon, 2015)

In this section the system requirements are described. These requirements are classified into groups of functional and non-functional and prioritized using the MoSCoW criteria.
Functional Requirements

1. The system must provide a Graphic User Interface.
2. The system must have the option to load various training files.
3. The system must process format-free plain text files for training.
4. The system must clearly separate the training and the testing phases.
5. The system must provide a feedback option to the user in order to verify the accretion of the annotation.
6. The system must provide alternative labels to a given annotation.
7. The system must prevent that the user could assign an entity into unknown categories.
8. The system could allow only certain type of files to be uploaded as training files.

Non-functional Requirements

1. The training phase of the system should not take more than 30 seconds.
2. The testing phase of the system should not take more than 20 seconds.
Chapter 3. System Design

The approach to overcome the potential issues that conveys the supervised labeling process is mainly focused in reducing the participation of a person in defining the training dataset and in the reclassification phase of the process. The aim in this project is to obtain a piece of software whose training dataset could be any plain text and whose output corresponds to tagged annotations and includes an option to automatically reclassify the wrong annotations, hence, since the user’s point of view, the system could be modeled as a black box system, as illustrated in the following figure.

![Black box model of the system](image)

**Figure 2 Black box model of the system**

Components Interaction

In this section we are going to decompose the system considering the existent tools and the approach to be considered to implement the feedback of the user.

The first consideration is that the input of the system, regard to the training phase, are text without specific format, then in this case the best alternative that fits this description is a NER (Named Entity Recognizer), which takes text input and returns the entities found in the text and their corresponding labels.

Once this result is obtained, it should be used to train a sequence labeling tool, this means that this tool could make some computations in order to infer the corresponding labels to a new data. Is in this labeling tool where the system will consider the test dataset as an input, apply the model and return the annotated text. In this section, the training dataset must be defined in specific formats that depend on the implementation of the algorithm; hence it is necessary to define an intermediate data structure that will be used as training data set of the sequence labeling tool.

Regard to the feedback from the user, this will be limited to wrong/right options, and the strategy to include this information to adjust the model behavior is to modify the training dataset in order to perform the corrections needed.

To facilitate the interaction between the user and the system, a GUI must be implemented.
After the description of the different components that will be considered in this system, a basic diagram considering the components is defined in the following figure:

![System decomposition diagram](image)

**Figure 3 System decomposition**

The selection of each one of the pieces of software is described in the Implementation chapter, along with the rationales that support the elections done.

**Handling the user interaction with the system (Microsoft; Microsoft)**

The implementation of the system will manage the interactions performed by the user using the pattern Model-View-Controller. This pattern allows decoupling the responsibilities in an abstract perspective of layers, where each layer is responsible for handling different tasks, as briefly described:

1. **View Layer.** This is the direct interface to the user. This receives the interactions coming from the user and displays the results of computations. Is in this layer where the GUI is defined.
2. **Model Layer.** Responsible to execute the tasks and computations in order to attend a user request received via the controller layer. The result will be the sent to the controller layer.
3. **Controller Layer.** This layer is the connection between the view and the model; in the first phase of its functionality it receives a request from the view layer and queries the model defining the corresponding parameters. In the second phase it receives a result to the query sent to the model layer and updates the view accordingly to display the result.

The following figure is taken from the Microsoft site specialized in the documentation related to applications’ developers, and it clearly shows the relationships among the layers described in the previous paragraph.

![Model-View-Controller diagram](image)

**Figure 4 Model-View-Controller (Microsoft; Microsoft)**
**Package definition**

The separation into packages offers advantages related to the comprehension and the debugging process of the code. To define the packages that will constitute the system, I have considered the MVC pattern, conforming packages to group the classes accordingly to their functionality under this schema.

Other three packages will be defined as utilities and complements, the first correspond to the “constant” package that contains all the final variables that will be used at certain points of the project. The variables of this package are defined as public due to the fact that must be available to every package in the project. The second package is called “util”; this contains a set of defined methods that could be called at one or several points of the project, hence these methods are public to avoid accessibility issues. The third package is called “main” and contains the Main class where the GUI is launched into the main method.

The corresponding relationship between the different packages is shown in the following class diagram obtained using the plugin of Eclipse.

![Class Diagram](image)

**User interface Design**

The user interface consists in a frame containing areas for buttons with Train and Test functionalities, a menu titled as File, text area to enter the text to be tested and an area where the labelled data will be put into a format of a table, where the entities are clearly identified by the corresponding label. The last column of this table contains a box confirming the correctness of the annotation done. By default, this is marked, as we assume the annotation is correct. Generally speaking the interface satisfies the recognition principle stated by (Nielsen, 1995).

The interface appearance, done in the specialized software Blasamiq, will look like the following figure.
Figure 6 Graphic User Interface Design
Chapter 4. Implementation and Testing
To implement the software we have different alternatives.

On one hand we can decide implement each component needed for the operation tailoring the whole system according to the specifications. This approach gives to the developers the complete control of the system, as the implementation of the different algorithms depend on them. However, this perspective is increases the time needed to produce the final system.

On the other hand, we can reuse available code assembling the system upon individual independent pieces of software. Under this methodology the process of pairing the different system components could be unfeasible and the documentation is a potential limitation of this process.

The points of view under which the system will be implemented correspond to a hybrid approach, i.e. use available APIs available for the components and define additional lines of code to fulfill the system requirements.

The components diagram that represents the interactions among the different components, including the specific names of those selected, that constitute the system is defined in the following figure.

![Components Diagram](image)

**Figure 7 Detailed components Diagram of the system**

**Rationale of the APIs selection**
The implementation of the project will include APIs that are available and that have appropriated license terms.

From the required research necessary to develop the project, the following rationalizations were derived to support each election of the APIs that will be considered.

**Election of ROSeAnn as NER provider**
The main advantage of ROSeAnn is that it implements the reconciliation algorithm described in (Chen, Ortona, Orsi, & Benedikt), this means that the annotation returned by the system have been already treated and assessed ensuring a high probability of assertion, as this annotation is the result of various annotations returned by black box NERs available online.
This tool has been set up to ensure that only fully operational web services are consumed by it. Additionally, we can also consider that the license terms allow the usage and modification of the source code, as long as the license terms are maintained.

**Election of CRF Project from sourceforge as sequence labeling tool**

The selection of the CRF algorithm is done considering that in the papers (Sha & Pereira, 2003) (Lafferty, McCallum, & Pereira, 2001), experimentally was determined that this algorithm returns better results when compared with traditional perspectives, such as Maximum Entropy Markov Models (MEMM) and other Hidden Markov Models (HMM) approaches.

From the research carried out as part of the project, the selected CRF API, available on the web (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Sourceforge) and whose complete documentation is also published (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Project Page), provides a stable and user friendly interface, which facilitates the usage and the interaction with this tool. Additionally, we can consider that the documentation available about this project is complete and useful. Finally, the license terms allow the use of the source code.

**General Overview of CRF**

Due to the fact that an API that already implements this algorithm is available, it is not necessary to understand the full details of the algorithm itself, however the general perspective of how it works and the basic definitions must be taken into account.

The following figure represents these perspectives; this figure has been taken from (Arranz, 2011), page 20 figure 2.1.

![Linear Chain Comparison](image)

**Figure 8 Linear Chain Comparison**

At the left we see the representation of HMM. In this scenario a given label (y) depends on the previous label and that each observation(x) only depends on the actual state of the labels.

The figure in the middle represents a MEMM. This is similar to the HMM perspective, differing in the fact that each label (y) will depend on the previous label and also in a vector of observations (X).

The figure in the right location shows the CRF perspective to overcome some of the main limitations that arise in HMM and MEMM approaches and it consists in considering the probabilities of a label (y) is conditioned to the immediate adjacent labels, rather than in the joint probability of the labels upon each one of the observed (Lafferty, McCallum, & Pereira, 2001). This also considers the vector of observations (X), similarly to the MEMM approach.
Usage of selected APIs

This fragment of the dissertation will describe in general terms the procedures taken to use the selected APIs. A detailed specification of the APIs could be found in the documentation related to each one, which is posted online at (DIADEM) and (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Project Page). This descriptions detail a general approach as well as the most significant decisions that could affect the accomplishment of either functional or non-functional requirements.

Use of ROSeAnn API

The ROSeAnn API is divided by packages according to the functionalities and to the online NERs that it could be connected to via Internet. The configuration file that defines the online NERs that will be considered in the aggregation algorithm is contained in the conf folder of the project and is identified by the name TextannotatorConfiguration.xml. After running tests, it was determined that the only two NERs that still online and processing the requests done by the tool are illinoisner and stanfordner. The other online NERs could not be reached either because of a license is needed to consume the service, or the links are already modified by the providers of these services. This led us to modify the xml configuration file in such a way that the only two threads that are going to run are those related to the fully functional NERs.

Another aspect to consider was the timeout defined until the system stops waiting an answer to a petition sent to the NER, as stated in the requirements, the training phase of the automatic annotator system should not take more than 30 seconds, being that the justification of modifying this parameter in the same configuration file and tagged as 
\[\text{<timeout>10000</timeout>}\], where the timeout is specified in milliseconds.

The ROSeAnn API is wrapped into a new class defined by the name RoseAnnotator and included in the package controllers of the system. This was decided to facilitate the API manipulation and also define methods that will perform computations that will be used a posteriori.

Use of CRF

The consumption of the CRF API should be configured by following the steps defined in the documentation of the API, from these 5 points; the most important are the obligatory implementation of the interfaces DataIter and DataSequence. The first named interface will be used to accessing the training data and the second interface will be employed to encapsulate the datasets (training and testing).

The CRF implementation used in this project, considers only numerical integer labels, so an intermediate process of label mapping must be completed before the training dataset enters to the system.

Implementation of the feedback loop

To implement the feedback component, a piece of code is developed and it will take binary feedback (i.e. right/wrong selection) from the user. In case that the user explicitly marks an error in an annotation, the system will modify the artifact used as trainer at the input of the CRF system, then a new training and testing phase is triggered.
The modification of the training artifact will take into account the output of ROSeAnn, establishing the frequency of occurrence of the different labels and then adding to the training dataset of the CRF system a new entry where the entity is the same as the mistakenly annotated and the new label is the most probable from the training dataset.

**Rationale of the Data structures selection**
Prior to the definition of the data structures, we should consider where these structures are going to be used and what the purposes of each one of them are, what operations should it support and how we can ensure an adequate time complexity.

**Data Structures related to CRF API**
In first place, the use of the CRF API requires a mapping process to be done between string labels and integer labels, such that each label corresponds to an integer. Bearing in mind that the number of labels is undefined beforehand, the assignment of the integers to be used to map each label must be done dynamically. Also the data structure that will store the labels that are progressively found needs to support dynamic growth, thus the most appropriated structure to store the label objects is an Array List, where the size of this structure will be increased each time a new label is discovered and assigned as code of the new label, ensuring that the mapping process will be done correctly. In this point we are assuming that, regardless the amount of labels that the annotator finds, that the text will always contain at least one label identified as NOTHING and its corresponding code 0.

To train the model upon the output that ROSeAnn returns, we define the object LabeledDataModel in the Model package. This object will then be the type of data contained in the DataSequences class of the controllers package, that should implement the DataSequence interface, as stated in the API documentation. (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Project Page). Then the data structure that will implement the artifact using to train the CRF API corresponds to an Array List of objects of type DataSequences and is declared in the class MainWindowController of the controllers package as the field trainingList.

**Data Structures related to the feedback section**
For the implementation of the system, we are going to define a data structure that keeps control of the labels that have been already discarded by the user as correct labels. The intention of this data structure is to control the labels that are found as the user provides feedback of value wrong to the same word, we are not focused counting the amount of times that a given label appears, and therefore, the most logic selection of the data structure is a Set, as this does not allow duplicated objects. Additionally, as the order in of the elements is not important for further computations, we can use HashSet as the data structure; this offers the advantage of fast access to the information, hence lower latency. This data structure is defined as the field usedLabels of the class LabeledDataModel located in the package model.

**Integration approach in the project**
The Integration part of the project, consist in connecting ROSeAnn and the CRF API in cascade. This means that the output of ROSeAnn will be used as the input of the CRF API.

To achieve this connection, an intermediate artifact is generated, this artifact will be assembled upon the output of ROSeAnn and will beformatted as required in the
documentation of the CRF API selected; this phase will be done automatically, and this artifact is not available for the direct manipulation of the user, ensuring that only defined categories are going to be considered for the corrections.

Feedback approach in the project
The effect of the user’s feedback takes place in the trainningList field of the MainWindowController of the controllers package. The modification that will occur is the addition of an object of type DataSequences to the trainningList field, where the new training dataset will contain the same dataset previously used with the addition of a new entity that consists of the word marked as mistakenly annotated and the new label will be the most probable label determined based on the training datasets.

Graphic User Interface Implementation
The implementation of the GUI is done using the Swing Java Library, where we can define the different components.

The GUI appearance implemented is shown in the following figure.

![Figure 9 Graphic User Interface Implementation](image)

The main frame is titled as Automatic Annotator, allowing the user to identify which is the main functionality of the program.

At the top of the panel the File menu, the “Train” button located at the bottom of the panel will be used in order to load the training files and The “Test” button will use the model learned from the training sets and apply it to the plain text that the user defines in the “Testing Text” text area, located at the left of the panel.

In the table located at the right of the frame, the system will display the results of the annotations for each word of the “Testing Text” area. A complete version of the user’s guide is included in the Appendix A of this document.
Testing strategy

The strategy to be used is to assess the most important parts of the system and to assess the system as a whole. It is important to consider that the system includes pieces of software that convey some level of randomness.

The system will be subjected to white box testing and black box testing.

White box testing

In order to summarize and present the tests to be applied to assess the system, we have adapted the table in the page 5 contained in the document available online at (TMap Tests Template).

Explanation of the following table:

<Test level> prioritizes the tests according to the importance of the feature to be tested. If it constitutes an important feature, more intensive must be the test to run.

<table>
<thead>
<tr>
<th>Characteristic/object part</th>
<th>&lt;Test level&gt;</th>
<th>Test type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training phase time</td>
<td>●●</td>
<td>&lt;&lt;Performance test&gt;&gt;</td>
</tr>
<tr>
<td>Testing phase time</td>
<td>●●</td>
<td>&lt;&lt;Performance test&gt;&gt;</td>
</tr>
<tr>
<td>Intermediate artifact to train the CRF</td>
<td>●●●</td>
<td>&lt;&lt;Integration test&gt;&gt;</td>
</tr>
<tr>
<td>User-friendliness</td>
<td>●</td>
<td>&lt;&lt;Usability test&gt;&gt;</td>
</tr>
<tr>
<td>Feedback loop functioning</td>
<td>●●●</td>
<td>&lt;&lt;Functional test&gt;&gt;</td>
</tr>
<tr>
<td>Training file type limitation</td>
<td>●</td>
<td>&lt;&lt;Functional test&gt;&gt;</td>
</tr>
</tbody>
</table>

Table 1 Testing Plan

To execute the testing tasks defined, we are going to use the experimental dataset included in the downloand of the mallet project (UMASS). To perform time measurements of the training and testing phases of the system, a statement to acquire the current time of the system at the beginning and at the end of each phase, subtract them and then determine if it is within the expected threshold defined in the non-functional requirements.

Black box testing

In this section of the test will be conducted from the point of view of a final user. This segment of the testing will contain images of the system being used. To perform these tests, the same sample datasets will be used.
Chapter 5. Project Management

To obtain the final system, the first step of the project is to identify the feasibility of the execution of the project. In order to do this, we performed a review in the available literature related to the theory of the project.

After the review of literature was finished, it was determined that it was necessary to perform the integration of two pieces of software that are already defined as APIs and to implement an additional piece of code that will modify the artifact used for the integration between the APIs. Once identified the two macro tasks that should be performed to obtain the final product, the project was organized in micro tasks to ensure the normal workflow.

The project management, in terms of time management is summarized in the following section.

1. Project proposal. 2 weeks.
2. Literature review. 2 weeks.
3. APIs selection. 1 week.
4. APIs’ testing. 1 week.
5. System design. 2 weeks.
6. Integration implementation. 2 weeks.
7. Feedback implementation. 2 weeks.
8. User interface implementation. 1 week.

It was necessary to perform some tests in the APIs to define the correct method to integrate the existent APIs and the most appropriated approach to implement the feedback section.

During the implementation of the project, it was crucial to have a complete documentation of the APIs which facilitate the implementation of the software.

From the perspective of process model used in the project, it was the waterfall model approach and component-based software engineering which were used, as described in (Bahsoon, 2015), since each phase of the implementation was performed after the previous one was completed, and the system implementation is done upon two APIs that are available online which are ROSeAnn (DIADEM) and CRF sequence labeling tool (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Sourceforge). To ensure that the system is correctly designed and implemented, it was necessary to clearly define the requirements, i.e. what is the purpose of the system and how this piece of software will be used.

The waterfall approach was useful for the realization of this project, however once that the software is already implemented a different approach could be considered to improve the system’s automation in the learning process. The resultant piece of software could be considered as a first generation product that could be engineered considering an evolutionary approach to ensure that the system behavior fulfills the users’ expectations.
Chapter 6. Results and evaluation

Results of testing
In this part, we describe the actual tests run to test the accomplishment of the specifications of the system.

To ensure that the requirements related to system’s performance are met, here a set of six tests are run. Then, the value that we will consider to check the accomplishment of performance specifications will correspond to the average value in the response times.

White box tests results
In order to quantify the time of response, some lines of code were included, as is shown in the following figures.

![Figure 10 Code to measure the performance of the training phase](image1)

![Figure 11 Code to measure the performance of the testing phase](image2)

The corresponding lines of code could be identified by the comment “//To measure performance”. It is important to notice that the elapsed time returns a value expressed in milliseconds.

The following figures illustrate the results of the tests. Here we are running a set of six trials on the training and the testing phases independently; therefore the results are considered and shown in different sections.

Results of the Training phase

```
Time of training phase is: 25220ms
```
Figure 12 Training phase results

Results of the Testing phase

Time of training phase is: 26996ms

Time of training phase is: 25543ms

Time of training phase is: 23915ms

Time of training phase is: 26763ms

Time of training phase is: 23533ms
Results of the training artifact

The training artifact is defined to be an Array List of objects of type DataSequences, this consists in a defined type used to be processed by the CRF API selected for the labeling tool.

In order to test that this artifact is being generated in the correct manner, we included an echo in the code after this Array List is obtained upon the output of the ROSeAnn API, this is shown in the following figure.

A fragment of the output is shown in the next screenshot.

This is a partial segment because the whole artifact obtained contains as much elements as words contains the training files used.
Results of the feedback loop

The feedback loop should be able to assemble a new object of type DataSequences in the training dataset and retrain the system again. As the data structure used for implementing the artifact is an ArrayList, the new object should be included as the last entry of the Array, i.e. must be the element size-1, where size is the size of the array with the new entry.

To test this fact, we added a line in the code to print the new object in the console and visually verify it corresponds to the object.

```
    public ArrayList<LabeledDataModel> trainAndTestModel() throws Exception
    {
        ArrayList<LabeledDataModel> labeledData = null;
        if (tableLabelsModel != null && tableLabelsModel.getLabeledData().size() > 0)
        {
            ArrayList<LabeledDataModel> newTraining = new ArrayList<>();
            for (LabeledDataModel label : tableLabelsModel.getLabeledData())
            {
                if (!label.getLabelCorrect())
                {
                    newTraining.add(new LabeledDataModel(label));
                    label.setLabelCorrect(Boolean.TRUE);
                }
            }
            if (newTraining.size() > 0)
            {
                DataSequences newTrainingSequence = new DataSequences(newTraining);
                trainingList.add(newTrainingSequence);
                System.out.println(trainingList.get(trainingList.size()-1)); //echo to test the feedback implementation
            }
        }
    }
```

Figure 16 Code to test the feedback segment

In the test we run, we take just entity as example to verify the assembly of the object.

Figure 17 Feedback example

After the Test button is clicked by the user, the object is correctly generated and successfully added to the training dataset.

Figure 18 Entry created by feedback segment

The following table contains the summary of the results of the tests.
<table>
<thead>
<tr>
<th>Characteristic/object part</th>
<th>Input</th>
<th>Expected Output</th>
<th>Observed Output</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training phase time</td>
<td>Sample Dataset</td>
<td>&lt;30.000 ms</td>
<td>25.328,33 ms</td>
<td>Results averaged over 6 trials.</td>
</tr>
<tr>
<td>Testing phase time</td>
<td>Sample Dataset</td>
<td>&lt;20.000 ms</td>
<td>13.338,17 ms</td>
<td>Results averaged over 6 trials.</td>
</tr>
<tr>
<td>Intermediate artifact to train the CRF</td>
<td>Sample Dataset</td>
<td>ArrayList&lt;DataSequences&gt; trainingList</td>
<td>ArrayList&lt;DataSequences&gt; trainingList</td>
<td>The format of the objects is defined as feature vector.</td>
</tr>
<tr>
<td>User-friendliness</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Is easy to identify how to use the system.</td>
</tr>
<tr>
<td>Feedback loop functioning</td>
<td>Sample Dataset</td>
<td>Elizabeth#{}Label#{}numericLabel#{}0#{}9#{}true</td>
<td>Elizabeth#{}Person#{}1#{}0#{}9#{}true</td>
<td>In this case, the object Elizabeth was originally annotated as nothing, and now it can be verified that the entry is created in the training artifact.</td>
</tr>
<tr>
<td>Training file type limitation</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>This is verified visually.</td>
</tr>
</tbody>
</table>

Table 2 Tests Results Summary

**Black box tests results**

This section is intended to verify the system functioning from the user’s perspective.
In the third iteration of the feedback the annotations got the correct labels.

The advantage of this individual feedback is that the corrections could be provided individually and also, the correct label should be accomplished after little iteration.

**Evaluation**

To evaluate the product implemented, it is important to consider the main objective of the software, which is to reduce the supervision in the labeling process. In the context of the project, we can understand the supervision as the direct participation of a person in the process. In a formal and concrete manner, we find stated in the project proposal that “The aim of the project is to use state of the art sequence labeling models which traditionally use labeled dataset as an input, in such way that the new input will be black-box annotators and it will contain a feedback option that the user will employ in order to validate the results.” (Jimenez Illescas, 2016).

As has been shown in both in white-box tests as in the black-box tests, the NER used for this project is successfully connected to the sequence labeling tool selected, which implements the CRF algorithm. Regard this section, the aim of the project has been successfully achieved and tested. In relation to the feedback section implementation, we have shown in the tests that it is correctly implemented and it leads to a correct output to the system.

Considering the arguments described above and upon the project proposal, the objective of the project is fulfilled. This does not imply that the piece of software obtained could not be improved; in fact the waterfall model considers that, after the system is implemented and tested, the operation and maintenance phase could modify the implementation to include functionalities or optimize the system features.
Chapter 7. Discussion and conclusions

Discussion

Project Summary
A significant reduction of the supervision has been achieved. Specifically the reduction in the participation of the users are evidenced in the training phase, where the training sets could be plain text, and the only limitation is that the files must be in English and pure text format (.txt). Also the feedback is implemented in a way that the user could not arbitrary modify neither the datasets nor the labels in the results, instead the feedback from the user is bounded to binary options (right/wrong), and is the software the responsible of the computation to decide which is the most probable label that a given entity must have.

Possible improvements
The system relies in the well use of it from the user, i.e. inadequate feedback is not considered. In order to overcome this issue, one approach is to increase the redundancy of the dataset doubling the feature vectors in order to train the CRF API. Although this could reduce the probability of the introduction of bias in the system, this decision could significantly increase the time taken to train the model and also the space used for the artifact will also be affected.

Considering the possibilities of implementation of the system, the design of this tool could have been done considering other available APIs and assess the most appropriated approach in order to determine the most effective implementation as the one which requires less interaction from the user. Due to the limitation of time, was not feasible to perform the comparison among other implementations.

Potential users of the system
The Named Entity Recognition process is particularly interesting for information extraction chores. In the paper (Molla, van Zaanen, & Smith), a possible application of the NERs is described to reduce the supervision in tasks of Question Answering. As the system developed in this project consists in an improvement of the available NERs, coupling them to CRF API and including the user’s feedback, this could also be considered as a possible application of the obtained program.

Conclusions
The final product obtained accomplishes the main goals stated in the project proposal, this means, the participation of the user in the learning process is limited. This reduction could be noticed in the training and correcting process of the system. The system was successfully developed and satisfies the requirements defined in the documentation and the main goal stated in the project proposal.
References and Bibliography


Jaiswal, A., Tawari, S., Mansuri, I., Mittal, K., & Tiwari, C. (s.f.). *CRF Project Page*. Recuperado el 04 de 09 de 2016, de sourceforge: http://crf.sourceforge.net/

Jaiswal, A., Tawari, S., Mansuri, I., Mittal, K., & Tiwari, C. (s.f.). *CRF Sourceforge*. Recuperado el 04 de 09 de 2016, de CRF Sourceforge download: https://sourceforge.net/projects/crf/


Submitted code details
This section describes the parts of the code defined by me, the parts of the code automatically generated and the parts that were adapted, where applicable.

1. CRF Project. Contains CRF API downloaded and available at (Jaiswal, Tawari, Mansuri, Mittal, & Tiwari, CRF Sourceforge).
2. FinalAnnotator. The code contained in this project is defined by me, except the getters and setters of the classes that were automatically generated in the IDE, and in adapted in order to get a readable implementation.
3. DependenciesLocation.txt. specifies the address where all the jar files, compiled from the ROSeAnn project available at (DIADEM).
4. AppendixD_ExperimentalData. Contains all the experimental data used to perform the tests. This sample datasets are available in the download of the Mallet project (UMASS).
Appendix A. User’s Guide
The graphic user interface will be launched in the center of the screen. This will be seen as in the next figure

![Automatic Annotator](image1)

**Training the system**
In the training phase, the main aim is to load to the system any plain contain, as long as this format is represented in a text (.txt) files, it could be loaded. These files will be used by the system to teach the system how to distinguish the different labels. To begin the system’s training, please click on the Train button.

![Automatic Annotator](image2)

Select the training files.

![Automatic Annotator](image3)
The dialogue will show that the system is performing the computations of the model. Once finished, the system will appear in the same way that in the first screen.

**Testing the system**

To test the system, you should write put some plain text in the Testing Text area. This could be any piece of text that you want to get annotated and then click on the Test button.

**Provide feedback to the system**

After the Testing phase is finished, the system will return a set of annotations located in a format of table at the right of the screen.
Here we can also provide feedback to the system, where if the box is checked this means that the word is correctly labeled and if it is unchecked this implies that you disagree with the label given to the word. If you want to retrain the system considering the feedback given, please click on the Test button.

After the retraining is finished, a new label will be assigned to the word.

You can provide feedback to a given word repeatedly.
Appendix B. How to run the code

Before running the code, we have to consider the following aspects.

1. The IDE used for writing and running the code is Eclipse.

![Eclipse IDE](image1)

2. The project which contains the source code of the implementation is identified as FinalAnnotator.

![Project Explorer](image2)

3. The project CRF contains all the GUI code related to the API which implements the CRF algorithm.

4. Include in the build path of the FinalAnnotator project the compiled jar files from the ROSeAnn tool.

![Build Path](image3)

5. Ensure that the FinalAnnotator project includes the CRF project as reference, in order to use the CRF API.

![Project Explorer](image4)

To run the code, you can find the main class, responsible for launching the GUI, in the main package of the FinalAnnotator project inside the Main class.
Appendix C. SVN content explanation
The SVN address (https://codex.cs.bham.ac.uk/svn/projects/2015/jcj584/) holds the following data:

5. CRF Project. Contains CRF API, with all the classes and dependencies required for the implementation.
6. FinalAnnotator. The project which contains the final version of the implementation.
7. DependenciesLocation.txt. specifies the address where all the jar files, compiled from the ROSeAnn project, are uploaded.
8. AppendixD_ExperimentalData. Contains all the experimental data used to perform the tests. This sample datasets are available in the download of the Mallet project (UMASS).