UNIVERSITY OF TRENTO ITALY

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE

FINAL THESIS

Italian Named Entity Recognizer

Author:
Festo OWINY

Supervisor:
Prof. Alessandro MOSCHITTI
Co-Supervisor:
Antonio UVA

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science in Computer Science

in the

iKernels Group
Department of Information Engineering and Computer Science

March 27, 2018

Declaration of Authorship

I, Festo OWINY, declare that this thesis titled, "Italian Named Entity Recognizer" and the work presented in it are my own. I confirm that:

- This work was done mainly while in candidature for a graduate program at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

Abstract

This thesis describes a novel statistical named-entity recognizer for the Italian Language. Named Entity Recognition (NER) is an important part of Natural Language Processing (NLP) and a subtask of information retrieval, where one seeks to find expressions of special meanings in texts written in natural language that can then be classified into proper nouns such as; persons, organizations, locations, geo-political entity, date, address, etc. These entities hold the key information in texts. Similarly, NER can be referred to as entity identification, chunking or extraction. Its results are often used as a preprocessing tool for other NLP tasks such as; Question Answering, Machine Translation, Text Summarization, Language Modeling or entity linking. The task has particular significance for Internet search engines, machine translation, the automatic indexing of documents, and as a foundation for work on more complex information extraction tasks.

The construction of a named entity system faces numerous challenges: questions of portability and system performance; A practical Named Entity system will require to be frequently adopted to new bodies of text and languages. NER system is usually robust across multiple domains, since it is anticipated to be applied on a diverse set of documents such as news articles, historical texts, webpages, patent applications, etc. However, It is challenging to build a system which can be ported with minimal expenses with at-most high accuracy in these new domains/languages.

Several approaches have been adopted to meet the above challenges. It is now very possible to find systems that can perform well in three or more languages with low costs. This thesis work is greatly inspired by EVALITA; a periodic evaluation campaign of NLP and speech tools for the Italian language. EVALITA is an initiative of the Italian Association for Computational Linguistics (AILC) endorsed by the NLP Special Interest Group of the Italian Association for Artificial Intelligence (AI*IA) and by the Italian Association of Speech Science (AISV). It facilitates a shared framework where various systems and approaches are consistently evaluated.

This project aims at researching and implementing NER System on the Italian I-CAB and CoNLL2003 datasets. We performed several experiments with linguistic and gazetteer features extracted from literature and several knowledge bases using CRF++ classifier; a simple, customizable and open source implementation of Conditional Random Fields used for segmenting/labeling sequential data. We ported the system to Italian language and our result on this task was competitive with the best systems built for Italian Language.

Comparatively, we performed experiments with Structured Perceptron classifier using *Extended* feature set. The results of the alforithm is comparative with those of CRF++. It is competitive with modern learning algorithms such as support vector machines (SVMs).

NER for the Italian language is not available in the DKPro framework at the moment. CRF++ performs very well and has a simplified feature template mechanism for easy integration of new unigram/bigram features. Therefore, we embedded the CRF++ classifier in a UIMA Annotator so that it can work with the rest of the UIMA framework. The tagger is freely available on Github as an Apache UIMA component. Our competitors are *Tint* and *TextPro*. But Tint is not based on UIMA thus they can not be integrated using the Software project management tool, maven. TextPro is only available for research and commercial purposes.

Acknowledgements

I would like to express my sincere gratitude to all who have supported me in different capacities during the course of this project:

A special thanks goes to my supervisor Prof. Alessandro Moschitti and Co-Supervisor Antonio Uva for guiding me. You did a great job providing insights and feedback to my submissions at various stages of the thesis.

Thanks to the University Students' fellowship and mamma Helen Robinson (major intercessor) for always standing in the gap. Your love for God and prayers is a true arsenal to bring success on the face of the earth! KEEP ON PRAYING !! Thanks to our God and Lord Jesus Christ who both intercedes for us and answers our prayers.

Most of all, I thank my greatly beloved son Benjamin Owiny who most inspires me to work harder. I always look forward to seeing him. Thanks to his mamma Victoria Nthuli for taking good care of him.

My father, Mr. Charles Boniface Epila Opio is a key role player to my success. He laid the best foundation in my life in Math, English, Chess and Scrabble amongst many. His sacrificial love as a father is loud and clear. My mother, Sophia is an adorable woman who ignites love and honor. I look forward to being a blessing to her. Thanks to my brothers and sisters, rest of the family and friends for being very supportive.

Contents

List of Figures

List of Tables

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
NER	**N**amed **E**ntity **R**ecognition
NLP	**N**atural **L**anguage **P**rocessing
IR	**I**nformation **R**etrieval
SVM	**S**upport **V**ector **M**achine
NER	**N**amed **E**ntity **R**ecognition
IE	**I**nformation **E**xtraction
CRF	**C**ondition **R**andom **F**ields
SP	**S**tructured **P**erceptron
UIMA	**U**nstructured **I**nformation **M**anagenement **A**pplications
DL	**D**eep **L**earning
CNN	**C**onvolutional **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
CAS	**C**ommon **A**nalysis **S**ystem
HMM	**H**idden **M**arkov **M**odel
MEM	**M**aximum **E**ntropy **M**odel
NE	**N**amed **E**ntity
ML	**M**achine **L**earning

Chapter 1

Introduction

At the Department of Information Engineering and Computer Science, the iKernels group led by Prof. Moschitti carries out extensive research on machine learning for natural language processing, information retrieval and data mining. The group is enthusiastic about improving the state of the art in Information Search and Retrieval by enriching language models with structural representations and semantics. One of the research areas is Named Entity Recognition with the goal of identifying and classifying proper names appearing in texts. This master thesis is a part of ongoing research in the field of information retrieval and has featured in various projects such as EVALITA and CLiC-it.

The legal domain as well as the media domain (classic newspapers or television (TV) news) both heavily rely on text as a central medium. These two media have been the most dominant consumption amongst humans. Nowadays, a large fraction of this information is consumed through internet offerings from social media platforms such as Facebook, Twitter, Instagram and others. Social media has suddenly dominated, disrupting the entire media industry.

Natural Language Processing applications are characterized by interdependent complex decisions requiring enormous amount of prior knowledge. In this report we investigate one of these applications, Named Entity Recognition (NER). Named entities range from simply singular words/entities (persons, locations and organizations) to chunks of text (e.g. *The Central Bank of Australia* or *University of Maryland Baltimore County*). Consequently, chunking model is required to predict whether a group of tokens belong in the same entity.

Named entity recognition is an important task of information extraction. Several work on named entity recognition has been carried out especially for English. In English Named Entity, the recent progress has been greatly facilitated by the MUC-6 and MUC-7 machine understanding conferences (Chinchor and Robinson, 1997). These conferences are initiated and financed by DARPA (Defense Advanced Research Projects Agency). They have the evaluation of information extraction systems as their primary purpose in a well controlled test which is followed up with a conference where several participants present their papers discussing the various methods incorporated (Borthwick and Grishman, 1999).

An earlier important question at the initiation of the NER task was whether machine learning techniques are necessary at all and whether simple dictionary lookup (Gazetteers) would be sufficient for good performance such as the CoNLL03 shared task. NER systems have been created that use linguistic grammar-based techniques as well as statistical models such as machine learning. Hand-crafted grammar-based systems typically obtain better precision, but at the cost of lower recall and months of work by experienced computational linguists. Statistical NER systems typically require a large amount of manually annotated training data. Semi-supervised approaches have been suggested to avoid part of the annotation effort. Many different classifier types have been used to perform machine-learned NER, with conditional random fields being a typical choice.

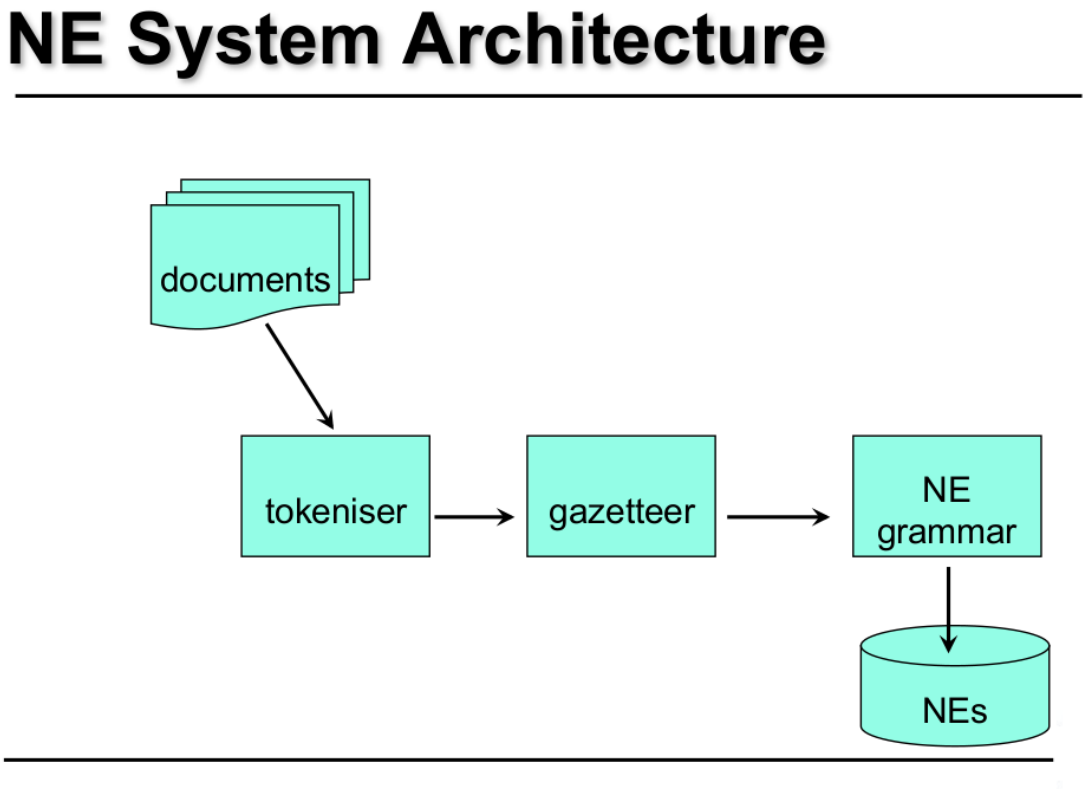

FIGURE 1.1: Common Named Entity Recognition System Architecture

1.1 Basic NE System

Nowadays, NER systems are becoming more and more complex. The Basic system consists of a pipeline of Unix process. Recognizing names based on character patterns (dates, numbers), recognizing names based on pre-stored names, applying mark-up to potential components of complete names, and recognizing names based on patterns of components.

A few of the patterns used require some additional context before a name is recognized. Patterns are then used to join together components on final pass and is made to recognize names of entities. Figure 1.1 illustrates a simplified NE system architecture; each component implements interfaces defined by the framework and is self-describing. How to combine the different components in a pipeline to build the desired NER system is an important subtask.

1.2 Thesis Outline

The thesis is structured as follows:

- Chapter 1: Introduction to the thesis topic providing motivations for the NER task.
- Chapter 2: The basic theory containing definitions of IR and NER, methods employable, applications of NER; the evaluation metrics used; detailed description of the algorithms used in this work to build the classifier and the background information required.
- Chapter 3: Describes the previous research works on the NER task. It provides a short description of various approaches used by earlier researchers.
- Chapter 4: Presents the proposed work with its implementation and the experimental results obtained and provides conclusion.
- Chapter 5: Describes wrapping the project in UIMA pipeline.

Chapter 2

Theory

2.1 Named Entity Recognizer

2.1.1 Information Extraction

Information extraction (IE) systems is the task of automatically extracting specific kinds of information from documents as opposed to more general task of extracting all information contained in the document. Structured information is extracted from unstructured and/or semi-structured machine-readable documents. The goals of the system are: organizing information to make it useful; precisely put information to facilitate further inferences by computer algorithms. IE systems attempt to extract clear, factual information (such as; Who did what to whom when?). Typical subtasks of IE include: Named entity recognition, Coreference resolution, Relationship extraction, Semi-structured information extraction, Language and vocabulary analysis, Audio extraction, etc. IE has been the focus of the MUC conferences and more need for developing IE systems helping to cope with the big volumes of available online data.

2.1.2 Named Entity Recognition

NE a subtask of IE, involves identification of proper names in texts, and classifying them into a set of predefined categories of classes:
Three universally accepted categories: person, location and organization Other common tasks: recognition of date/time expressions, measures (percent, money, weight etc), email addresses etc. Other domain-specific entities: names of drugs, medical conditions, names of ships, bibliographic references etc. Named entity recognition, the subject of this thesis, is one of the simplest subtasks of IE tasks. Therefore a relatively high accuracy is expected of NE systems.

As previously defined a NE task is divided in two parts:

- Recognizing the entity boundaries.
- Classifying the entities in the NE categories.

While it is fairly easier to build a named entity (NE) system with a reasonable performance, there are still numerous ambiguous cases which make the task difficult to attain human levels of performance. For example;

- *Ms. Victoria Nthuli lost* 25 *pounds ...* Did she lose 25 pounds of weight or 25 pounds of British currency?
- When is the word *Israel* being used as the name of a person and when as the name of a country?

The ambiguity suggests a more general question of robustness and portability in NE systems: Does a NE system have to be rewritten whenever there is a shift in domain or language?

2.1.3 Methods for NER and IE

The following are the main approaches to NER and IE:

- Rule-based methods: Most of the MUC-7 participating systems used the *handcrafted approach*. The technique implies that the systems are built by hand and rely heavily on the human designers' intuition. This approach suffers slow and expensive computational linguist out of the development loop as described below;
  Given the usual time constraints, It is generally impossible to code for every exception which one can think of, leaving aside those exceptions which don't become apparent until one has run a test. Expense is another problem faced with the handcrafted approach whereby getting the system up and running requires several programmer time with significant experience in computational linguistics. These kind of specialists are a scarce leading to heavy costs. Then if we want to port the system to a new domain or language, this time is largely wasted. Another problem experienced is the issue of consistency and reproducibility of results. Finally, this approach despite enjoying high precision suffers low recall.

- Probablistic modeling and machine learning

  - Using classifiers (Generative: Naïve Bayes and Discriminative: Maxent models)
  - Using Sequence models (HMMs, CMMs/MEMMs, CRFs, SP)

  To solve the above problems of handcoded systems one is required to build a system which will automatically train itself thereby eradicating out of the development loop the slow and expensive programmers and computational linguists. The assumption that the data points are independently and identically distributed (i.i.d.) is strongly considered in Naïve Bayes and Maxent models. However, by moving to a scenario of structured prediction (such as HMMs, CMMs/MEMMs, CRFs), where the inputs are assumed to have temporal or spacial dependencies, we ignore the i.i.d. assumption.

- Deep Learning (Recurrent Neural Networks, Convolutional Neural Networks): Attributes are: Big training corpus, No feature generation, Defining the model, Training and inference. Why do we need to study DL in NLP? Provide state-of-the-art performance in many tasks (Example: machine translation), This is where most of research in NLP is now happening (Example: papers from ACL, EMNLP, etc.), Look fancy and everyone wants to know them.

Why do we need to study traditional NLP methods over the Deep learning approach ?

- Traditional methods Perform good enough in many tasks (Example: sequence labeling)
- Traditional NLP methods can help to further improve Deep Learning models (Example: word alignment priors in machine translation)
- Traditional NLP methods are more interpretable than Deep Learning approach
- Traditional NLP methods allow us not to be blinded with the hype (Example: word2vec / distributional semantics)

2.1.4 Applications of NER

Named Entity Recognition is a process where an algorithm takes a string of text as input and identifies relevant nouns (person, location or organization) mentioned in that string. These entities hold the key information in texts an the results are often used as a preprocessing tool for other NLP tasks. Accurate named entity systems have numerous benefits including the following:

- Classifying content for news providers: NER is able to scan entire articles and identify the key attributes (people, organizations and places) being discussed in them. This information helps in automatic categorizing of the articles in defined hierarchies enabling smooth content discovery.
- Efficient Search Algorithms: In designing an online search algorithm with numerous articles, It takes a long time for the algorithm to search all words in the multitudes of articles. NER is used to extract the relevant entities from the articles. Here, the search process speed up is given by the fact that search term is matched with only a small list of entities from each article.
- Question Answering (QA): The NER is embedded in a QA system and the overall QA system performance is better. For an example of how to use Named Entitities in Question Answering is well described in the academic papers *UIMA*severyn2013building and *UIMA*molla2007named.
- Machine Translation (MT): The results in *UIMA*babych2003improving show that combining NER with MT has a great potential in improving the state-of-the-art in output quality. A NE tagger can therefore serve as a preprocessing step for MT or boost its performance.
- General document organization: A user is able to call up all documents on a company intranet that mentions a particular individual.
- NE is a great preprocessing step for other NLP tasks such as *Language Modeling or entity linking* and *Text Summarization*.

2.2 Introduction to Machine Learning

Machine learning is a field of computer science that enables the computers to learn without being explicitly programmed. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data with ability to change when exposed to new data. A more formal definition by *Tom M. Mitchell* is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E. If we require our computer program to perform a classification task (T), we train it through a chosen machine learning algorithm with data from past classifications (experience E) and if it has learned the patterns or features in the data, it will do better at predicting the classes for unseen future object inputs (performance P). Effective machine learning is difficult because finding patterns is hard and often not enough training data are available; as a result, machine-learning programs often fail to deliver. Machine learning is closely related to computational statistics with strong ties to Mathematical optimization.

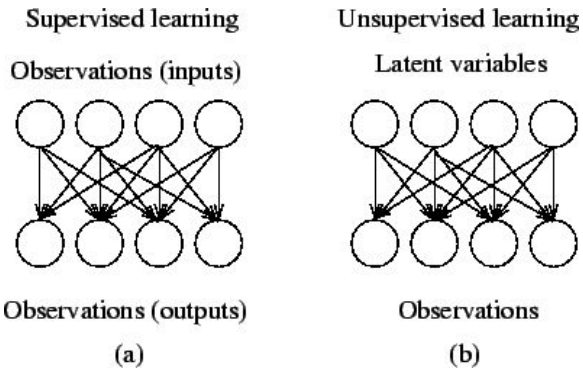

FIGURE 2.1: Supervised and Unsupervised learning

2.3 Types of Machine Learning Algorithms

Machine Learning algorithms are those that without human intervention are able to learn from data and improve from experience. The algorithms are grouped based on the learning style and they adopt in learning the model. Learning tasks may entail learning a mapping function, hidden structure in unlabeled data or 'instance-based learning. The two most important learning paradigms are as follows.

2.3.1 Supervised Learning

The computer is issued with labeled data as inputs and the corresponding outputs and the goal is to learn a general rule that maps inputs to outputs. The model is trained on this data to predict the label and a loss function corrects the model when these predictions are wrong. Supervised learning problems can be of two types: classification(where the objective is to predict the outcome of a given sample where the output variable is categorical) and regression(where the objective is predict the outcome of a given sample where the output variable is in real valued form). The goal of supervised machine learning is to use the training dataset D_L to learn a function(classifier) f that maps $X$to Y; Hence, given a new instance(test example) $x \in X$, the machine makes a prediction $\hat{y}$ by evaluating; $\hat{y} = f(x)$.

2.3.2 Unsupervised Learning

Unlike the supervised learning , unsupervised learning problems possess only input variables but no corresponding output labels. In addition, there is no loss function to correct the model since training data is unlabeled. The model is trained to detect structures and patterns in training data, thereby providing insights to the data. Unsupervised learning problems can be of two types: association(probability of co-occurrence of items in a collection) and clustering(grouping more similar samples together to form clusters). Here Machine learning models are trained from a set of observations only, $D_U := \{x^1, ..., x^M\} \subseteq X$

Figure 2.1 below illustrates the supervised and unsupervised learning methods.

2.4 Generative vs Discriminative approach

Assume a finite set of words $\sum$, and a finite set of tags $\bigwedge$. Define D_L to be the set of all sequence/tag sequence pairs $\langle x_1...x_n, y_1...y_n \rangle$ such that $n \geq 0, x_i \in \sum$ for $i = 1...n$, and $y_i \in \bigwedge$ for $i = 1...n$. Generative tagging model p is such that:

1. For any $\langle x_1...x_n, y_1...y_n \rangle \in D_L$,

$$p(x_1...x_n, y_1...y_n) \geq 0 \tag{2.1}$$

2.

$$\sum_{\langle x_1...x_n, y_1...y_n \rangle \in D_L} p(x_1...x_n, y_1...y_n) = 1 \tag{2.2}$$

Given a generative tagging model, the function from sentences $x_1...x_n$ to tag sequences $y_1...y_n$ is defined as below;

$$g(x_1...x_n) = \arg \max_{y_1...y_n} P(x_1...x_n, y_1...y_n) \tag{2.3}$$

Generative classifiers attempt to estimate the probability distributions $P(Y)$ (class prior) and $P(X|Y)$ (class conditionals). Examples are; *Naive Bayes*, *HMM*. Consider Generative Modeling (*HMMs*); Given training set X with label sequences Y: Train a model θ that maximizes $P(X, Y|\theta)$; For a new data sequence x, the predicted label y maximizes $P(y|x) = P(y|x, \theta)P(x|\theta)$

In contrast, discriminative models, also called conditional models, are a class of models that do not require modeling how the data was generated. It is used to model the dependence of target variables y on observed variables x. Examples of these classifiers are; *Perceptron*, *Margin Infused Relaxed Algorithm*, *Maximum Entropy* and *Support Vector Machines*.

Given discriminative model (for example *MEMMs*), training set X with label sequences Y, training a model θ that maximizes $P(Y|X, \theta)$. For a new data sequence x, the predicted label y maximizes $P(y|x, \theta)$. Disadvantages of MEMMs; $P(Y|X) =$ product of factors, one for each label and moreover each factor can depend only on previous label, and not future labels.

Therefore, the main difference between the two approaches is that generative classifiers model the probability distribution of the data $P(X, Y)$ whereas discriminative ones attempt to model the conditional probability of the sequence given the observed data $P(Y|X)$.

2.5 Conditional Random Fields

Several tasks involve predicting a large number of variables that depend on each other and also on other observed variables. Structured prediction methods are essentially a combination of classification and graphical modeling, combining the ability of graphical models to compactly model multivariate data with the ability of classification methods to perform prediction using large sets of input features *UIMA*mccallum2003dynamic.

In HMMs, to define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences. Since the inference problem for such models is intractable, it is not feasible to represent long range dependencies or multiple interacting features of observations. The main motivations for conditional models is attributed by the inference problem. A conditional model does not expend modeling effort on observations since It specifies the probabilities of possible label sequences given observation sequence. Transition probability between labels does not necessarily depend only on current observation but also on past and future observations. In contrast, in order to achieve tractability, generative models make independence assumptions on the observations, for instance conditional independence given the labels. CRFs is a sequence modeling framework with all the advantages of MEMMs but also able to solve the label

bias problem. Whereas CRF has a single exponential model for the joint probability of the whole label sequence given the observation sequence, MEMM models conditional probabilities of next states given the current state.

Conditional Random Fields (CRFs) (*Lafferty et al., 2001*) is a framework for building probabilistic models, segmenting and labeling sequence data, thereby offering numerous advantages over hidden markov models and stochastic grammars with the ability to relax the strong independence assumptions of these models (QUINTÃO, 2014). CRFs also avoid a fundamental limitation of maximum entropy Markov models (MEMMs), labeling bias, which makes them biased towards states with few successor states (Lafferty, McCallum, and Pereira, 2001). CRFs have seen wide applications in many areas such as; natural language processing, computer vision,and bioinformatics. Despite the fact that earlier applications of CRFs used linear chains, more recent applications of CRFs have used more general graphical structures. These recent developments are useful for predicting complex structures, such as graphs and trees and for relaxing the independence assumption.

CRF is a whole-sequence conditional model rather than a chaining of local models as shown in Equation 2.4. The space of $c's$ is now the space of sequences. But if the features f_i remain local, the conditional sequence likelihood can be calculated exactly using dynamic programming.

$$P(c|d,\lambda) = \frac{\exp \sum_i \lambda_i.f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i.f_i(c',d)} \tag{2.4}$$

Given a weight vector w, the conditional probability $P_w(Y = y|X = x)$ of label sequence given input sequence is defined in Equation 2.5. Both vector of local features (f) and corresponding weight vector (w) specify the *CRF* on (X, Y); In addition, each local feature exists either as a state feature $s(y, x, i)$ or a transition feature $t(y, y^{'}, x, i)$. Where $y, y^{'}$ are labels, x, i are input label and input position respetively. In CRF we assume the model decomposes into parts (*edges* and *nodes*) rather than computing the posterior marginals for the exponential output variables configurations. Therefore exploiting the decomposition and updating the "local features" only at the *nodes* and *edges* as above.

$$P_w(Y = y|X = x) =$$

$$\frac{1}{Z(w,x)} \exp(w.f_{init}(x,y_1) + \sum_{i=1}^{N-1} w.f_{trans}(i,x,y_i,y_{i+1}) + w.f_{final}(x,y_N) + \sum_{i=1}^{N} w.f_{emiss}(i,x,y_i)) \tag{2.5}$$

$f_{init}, f_{trans}, f_{final}$ and f_{emiss} are the initial, transition, final and emission local features respectively. $Z(w, x)$ is the normalizing factor, also called the partition function.

$$Z(w,x) =$$

$$\sum_{y \in \wedge^N} \exp(w.f_{init}(x,y_1) + \sum_{i=1}^{N-1} w.f_{trans}(i,x,y_i,y_{i+1}) + w.f_{final}(x,y_N) + \sum_{i=1}^{N} w.f_{emiss}(i,x,y_i)) \tag{2.6}$$

The optimization problem below corresponds to CRFs yielded by maximizing conditional log-likelihood from a set of labeled data $\{(x^m, y^m)\}_{m=1}^{M}$. It is regularized with Euclidean norm function to avoid overfitting

$$\hat{w} = arg \max_w \sum_{m=1}^{M} \log P_w(Y = y^m|X = x^m) - \frac{\lambda}{2} \parallel w \parallel^2 \tag{2.7}$$

TABLE 2.1: *CRF* implementations

	Link
CRF++	http://crfpp.sourceforge.net/
MALLET	http://mallet.cs.umass.edu/
GRMM	http://mallet.cs.umass.edu/grmm/
CRFSuite	http://www.chokkan.org/software/crfsuite/
FACTORIE	http://www.factorie.cc

During decoding three problems are dynamically solved; Given $X = x$, compute the most likely output sequence $\hat{y}$ that maximizes $P_w(Y = y|X = x)$
At each position i compute the posterior marginals $P_w(Y_i = y_i|X = x)$
Evaluate the partition function $Z(w, x)$. Training is slower, but CRFs avoid causal-competition biases.

Essentially, CRFs are extensions of the logistic regression classifier to arbitrary graphical structures. It is also as a discriminative analog of generative models of structured data, such as hidden Markov models previously discussed. Some of the implementations of Conditional Random Fields and the related links is shown in the Table 2.1

2.5.1 CRF++

CRF++ (Kudo, 2005) is an open source and easily customizable implementation of CRFs for labeling sequential data. NLP tasks such as Named Entity Recognition, Information Extraction and Text Chunking can all be handled with CRF++ given its general purpose design. CRF++ has several features that enhances its capability: It can redefine feature sets with fast training capability based on LBFGS algorithm and ability to perform single-best MIRA training. It also uses less memory during both training and testing with encoding/decoding roles. Further more, it can perform n-best outputs with extra ability to display the marginal probabilities of these candidate values.

CRF++ (Kudo, 2005) software, written in C++ with ST, is freely attainable as binary for Windows machines, or source code easily compilable on any machine with a c++ compiler from the *URL* https://taku910.github.io/crfpp/.

Inorder for CRF++ to work properly both the training and test files consisting of tokens are orderly formatted: A token consisting of fixed number of multiple columns usually corresponds to a word then a sequence of tokens formulates a sentence. The token columns are seperated by white spaces while two consecutive sentences are spereated by an empty line. Every column has similar type of item such as 'word', 'POS tag', 'PREFIX', et cetra; with the last column representing the answer tag to be trained by CRF.

In Table 2.2 there are 4 columns for each token: The word itself (e.g. German), part-of-speech associated with the word (e.g. JJ), Chunk tag (I-NP) and Named entity recognition (answer) (e.g. I-MISC); both represented in IOB2 format. The test data is formulated in a similar pattern except that It does not contain the NER (answer) column.

CRF++ command crf_learn collectively with the prepared template file are supplied with four parameters to control the training condition. The test phase entails crf_test command, trained model together with the test data and test parameters to control test output.

CRF++ being able to solve different tasks requires the feature templates to prepared in advance. The feature template describes the features used for training and testing with every line denoting one template as demonstrated in Tabel 2.3. Setting the word *'British'* as the reference point *(0,0)*, the template coressponds to expanded features: *boycott*, *British*,

TABLE 2.2: extract from training data CoNLL shared task

German	JJ	I-NP	I-MISC
call	NN	I-NP	O
to	TO	I-VP	O
boycott	VB	I-VP	O
British	JJ	I-NP	I-MISC
lamb	NN	I-NP	O
.	.	O	O
Peter	NNP	I-NP	B-PER
Blackburn	NNP	I-NP	I-PER
BRUSSELS	NNP	I-NP	I-LOC
1996-08-22	CD	I-NP	O

TABLE 2.3: feature template corresponding to Table 2.2

```
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[0,1]
U05:%x[1,1]
U06:%x[0,2]
U07:%x[-1,0]/%x[0,0]
U08:%x[0,0]/%x[0,1]
```

lamb, JJ, NN, I-NP, boycott/British, British/JJ respectively. The two template types; unigram and bigram are identified by the first character '**U**' and '**B**' respectively.

2.5.2 Reranking

In these approaches the output of an existing probabilistic parser (generative model such as CRF and SP) can be combined with a discriminative model, i.e. a reranker, to further improve its performance. For each input sentence, the base parser produces a set of candidate parses with associated probabilities that define an initial ranking of these parses. Using additional features of the tree in the second model, the initial ranking is improved. Reranking Strategy involves first training the CRF model to generate n-best candidates per sentence, along with their probabilities. Each candidate is then represented by a semantic tree together with a feature vector. As a binary classification problem, one sets the reranking task as in the paper (**nguyen2012structurall**). In this work we did not focus on reranking.

2.6 Structured Perceptron

We describe structured perceptron (Collins, 2002) algorithms for training tagging models as an alternative to maximum-entropy models or conditional random fields (CRFs). The

algorithm relies on Viterbi decoding of training examples and very simple additive updates. The perceptron algorithm in practice is very easy to implement and It performs remarkably well in various tasks. The two characteristics positions the structured perceptron algorithm as a natural first choice to use/test a new problem or a new feature set. Running the structured perceptron algorithm to obtain a weight vector w that accurately classifies the training data. Given the weight vector w, the conditional probability $P_w(Y = y|X = x)$ of label sequence given input sequence is defined in Equation 2.5.

The algorithms are based on the perceptron algorithm (Rosenblatt, 1958) but we use the voted (averaged) versions of the perceptron described in (Freund and Schapire, 1999). The voted version perceptron algorithms have been shown by (Freund and Schapire, 1999) to be competitive with modern learning algorithms such as support vector machines (SVMs). However, they have previously been applied mainly to classification tasks and therefore it is not entirely clear how the algorithms can be carried across to NLP tasks such as NER, POS tagging or parsing.

2.7 Feature types

In ML and pattern recognition, a feature is an individual measurable property or attribute of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective and efficient algorithms in classification, pattern recognition and regression. Features are usually numeric, but structural features such as strings and graphs are used in syntactic pattern recognition. The concept of "feature" is related to that of explanatory variable used in statistical techniques such as linear regression. Feature engineering involves developing systems to extract or select features requiring experimentation of multiple possibilities and the combination of automated techniques () with the intuition and knowledge of the domain expert. Table 2.4, Table 2.5 and Table 2.6 are key feature levels in NLP tasks.

2.7.1 Local and Global Feature Vectors

Here is a brief description of the feature vector representations which are commonly used in maximum-entropy models for tagging, and which are also used for Structured Perceptron. The tagging problem is decomposed into sequence of decisions in left-to-right manner where at every point there is a *history*. A *history* is formally defined as a 4-tupule $\langle t_{-1}, t{-}2, w_{[1:n]}, i\rangle$ where $w_{[1:n]}$ is an array that specifies the n words of the input sentence, $t_{-1}, t{-}2$ are the previous two tags and the index of the word being tagged, i. The global representations used for structured perceptron are simple functions of local representations as in the paper *UIMA*collins2002discriminative

2.8 UIMA

Unstructured Information Management applications (UIMA) is a software architecture that supports the development, composition and deployment of multi-modal analytics of unstructured information (text, audio, video, images, etc.) to extract for end users relevant knowledge such as entities (places, persons, organizations), relations (lives-at, belongs-to).

Apache UIMA includes APIs and tools for creating analysis components such as: tokenizers, summarizers, categorizers, parsers, named-entity detectors etc. Furthermore, in analyzing unstructured content, UIMA utilize a diversity of analysis technologies comprising: Statistical and rule-based Natural Language Processing (NLP), Information Retrieval (IR), Machine learning, Ontologies, Automated reasoning and Knowledge Sources

TABLE 2.4: Word level features

Features	Examples
Case	Starts with a capital letter word is all uppercased the word is mixed case
Punctuation	ends with period has internal period Internal apostrophe, hyphen or ampersand
Digit	Digit pattern Cardinal and ordinal Roman number Word with digits
Character	Possessive mark, first person pronoun Greek letters
Morphology	prefix,suffix, singular version, stem common ending
Part-of-Speech	proper name, verb, noun, foreign word
Function	Alpha, non-alpha, n-gram lowercase, uppercase version pattern, summarized pattern token length, phrase length

TABLE 2.5: List lookup features

Features	Examples
General list	General dictionary Stop words (function words) Capitalized nouns (e.g., January, Monday) Common abbreviations
List of entities	Organization, government, airline, educational First name, Last name celebrity Astral body, continent, country, state, city
List of entity cues	Typical words in organization person title, name prefix, post-nominal letters location typical word, cardinal point

TABLE 2.6: LDocument and corpus features

Features	Examples
Multiple occurrences	other entities in the context Uppercased and lowercased occurrences Anaphora, co-reference
Local syntax	enumeration, apposition position in sentence, in paragraph and in document
Meta information	URL, email header, XML section bulleted/numbered lists, tables, figures
Corpus frequency	word and phrase frequency co-occurrences multi-word unit permanency

(e.g., CYC, WordNet, FrameNet, etc.). These analysis capabilities though costly helps bridge unstructured to structured worlds. This is illustrated in Figure 2.2.

UIMA is a framework developed by IBM in the context of the Watson project. It is used for building complex pipelines able to process text in natural language e.g. English, Italian, etc. It allows complex pipelines to be decomposed in many pieces.e.g. segmentation, tokenization, lemmatization. UIMA building blocks are composed of :

- **Annotator**: extract structured information from text in form of annotations. e.g. tokenizer, pos-tagger
- **Annotation**: A structured object corresponding to a span of the original text.e.g. a token, a part-of-speech, a named entity
- **CAS**: An object containing;
  1. the original text to be processed; and
  2. the annotations returned by the annotators.
- Many UIMA annotators already available off-the-shelf thanks to the DKPro project
- Unfortunately, only few basic components available for the Italian language i.e. sentence segmenter, tokenizer, lemmatizer, pos-tagger
- Many other missing, e.g. NER
  - alternatives: Tint (not based on UIMA) and TextPro (commercially licensed).
- In this project, we wrote a UIMA annotator that wraps the Italian NER available on github

UIMA software architecture is composed of data representations, component interfaces, design patterns and development roles for describing, creating, composing, discovering and deploying multi-modal analysis capabilities. The UIMA framework provides a run-time environment where developers plug in their components and compose UIM applications. Apache hosts both Java and C++ implementation of UIMA Framework hence the framework is not IDE/platform specific.

The UIMA Software Development Kit (SDK) includes the UIMA framework, Eclipse-based (http://www.eclipse.org/) development environment supporting tools (Component Descriptor Editor and JcasGen) and utilities; extremely useful for orientation to complex interfaces of annotation components.

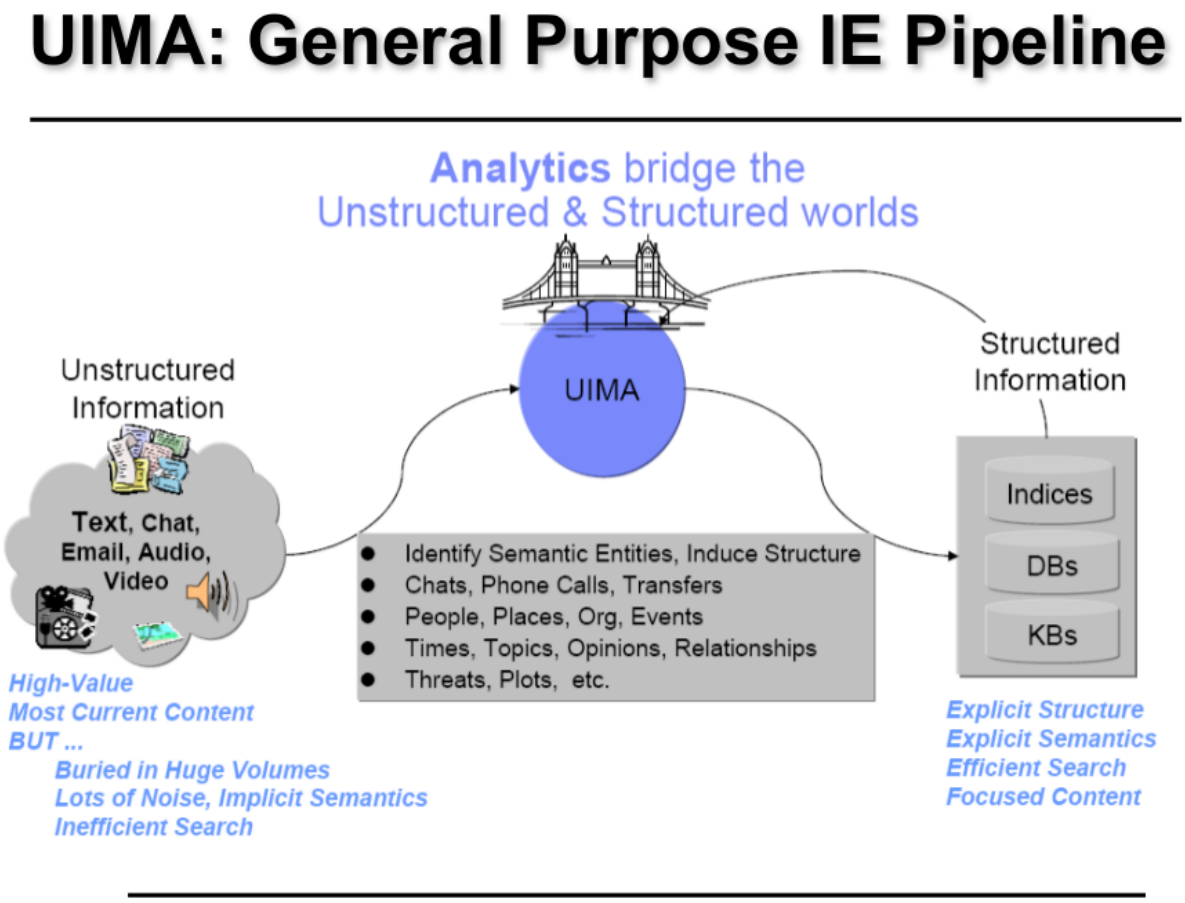

FIGURE 2.2: UIMA bridges unstructured and structured worlds

Apache UIMA is an Apache-licensed open source implementation of the UIMA specification as illustrated in Figure 2.3. Being an Apache project implies that It can be freely downloaded at http://uima.apache.org. (Previously, the architecture was proprietary to IBM).

UIMA framework provides the necessary methods for creating analysis engines which add the necessary APIs and infrastructure for the composition and deployment of annotators within the UIMA framework. Annotators represent and share their results with the Common Analysis Structure (CAS), an object-based data structure that allows the representation of objects, properties and values.

UIMA supports multiple views of a document enabling processing multiple forms of the artifact, e.g., the audio and the closed captioned views of a single speech stream, or the tagged and detagged views of an HTML document.

UIMA provides an efficient implementation of CAS with multiple programming interfaces to read and write analysis results and allow developer to obtain indexed iterators. Java CAS (JCAS) provides a natural interface to CAS objects in Java.

UIMA Context, the framework's resource manager interface, helps component developer to access external resources and ensures that aggregated annotators share same instance of an external file or remote resource accessed via its URL.

There are two parts to a component: declarative part (Component Descriptor) and code part (implemented algorithm); the code may be already provided in reusable subcomponents. The UIMA SDK provides tools (Component Descriptor Editor) for easily creating and maintaining the component descriptors relieving the developer from editing XML directly. The descriptor of a component serves as its declared interface to the rest of UIMA.

UIMA framework is equipped to manage different deployments, performing more complex tasks; the delegate engines, can be tightly-coupled (running in same process) or loosely-coupled (running in separate processes/different machines). UIMA supports several remote protocols for loose coupling deployments of aggregate analysis engines, including SOAP (Simple Object Access Protocol), a standard Web Services communications protocol.

The architecture also facilities wrapping components as network services, scaling up to very large volumes by running annotation pipelines in parallel over a networked cluster through a server that provides analysis results as a REST service. Flow Parallelization is enabled by user-provided flow controllers supported by the architecture.

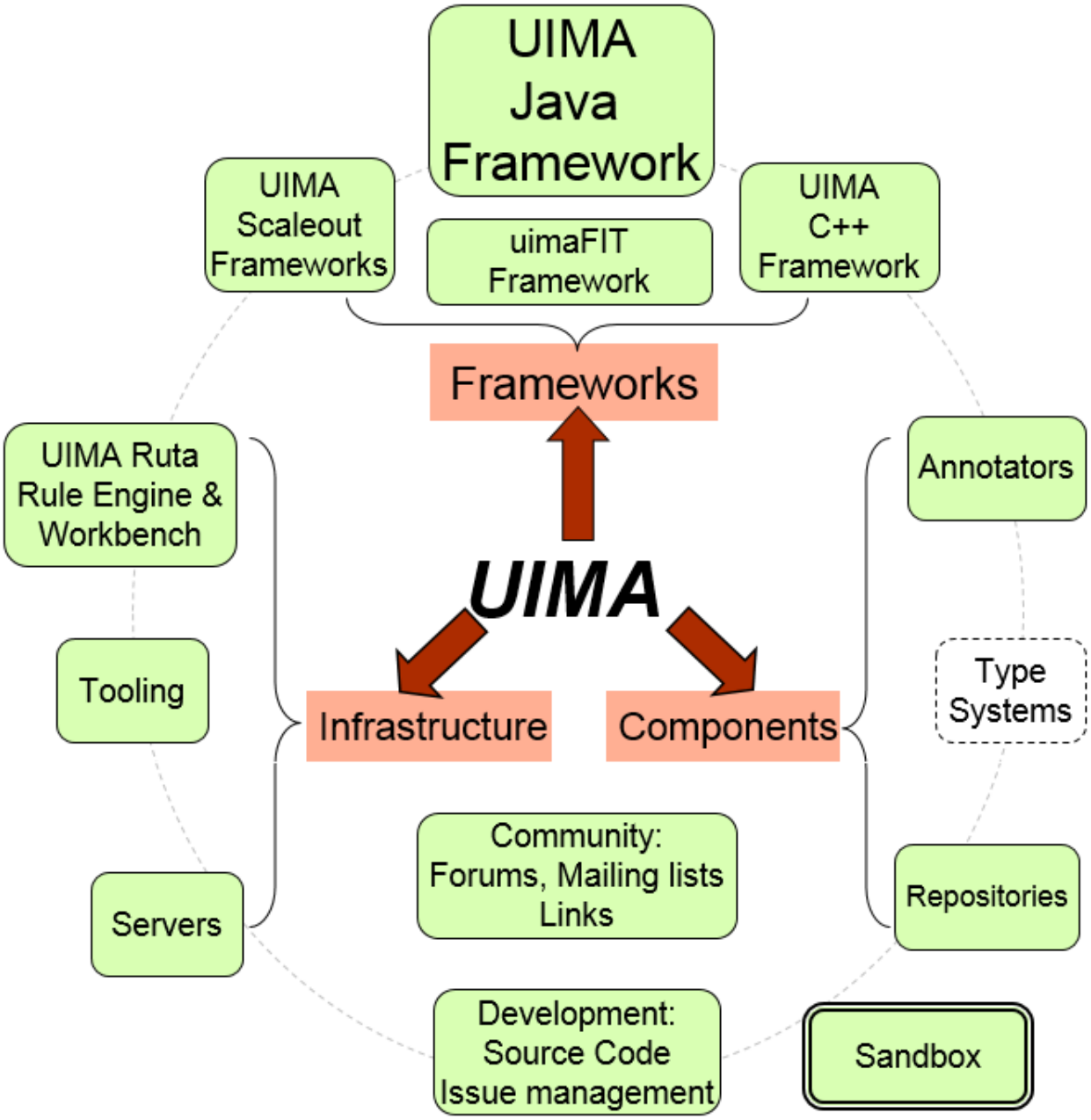

FIGURE 2.3: Apache UIMA Frameworks, Components and Infrastructure

Chapter 3

Related Work

3.1 Introduction

In this chapter, we discuss the studies so far made on NER for Italian Language. We give a brief description of AILC, EVALITA and CLiC-it. The performance score reported for these previous studies are for the I-CAB dataset which is the primary dataset used in this thesis work. It is notable that the projects are primarily extracts from EVALITA. Finally we present a table of results summary Table 3.1 for these related works and best systems.

3.1.1 AILC, CLiC-it and EVALITA

AILC is an Italian Association for Computational Linguistics bringing together different research groups and industries in Italy. The association aims to promote methodological, theoretical and experimental reflection, scientific cooperation, collaborations and technology/knowledge market transfer.

CLiC-it, an initiative AILC, aims to establish a reference forum for research on Computational Linguistics for Italian community. CLiC-it covers all aspects of automatic language understanding, both written and spoken, and targets state-of-art theoretical results, experimental methodologies, technologies, as well as application perspectives, which may contribute to advance the field.

EVALITA, another initiative of AILC is a periodic evaluation campaign of NLP and speech tools for the Italian language that facilitates a shared framework where various systems and approaches are consistently evaluated. EVALITA is endorsed by the Italian Association of Speech Science(AISV) and by the NLP Special Interest Group of the Italian Association for Artificial Intelligence(AI*IA). The aim of the campaign is to improve and support the development and dissemination of resources and technologies for Italian. The participating teams are from both academic institutions and industrial organizations. Five EVALITA evaluation campaigns have been held in 2007, 2009, 2011, 2014 and 2016 respectively. A recent call has been made for the 6^{th} evaluation campaign EVALITA 2018, to be held in Turin (December 10-12, 2018).

The evaluation campaigns are usually held as a co-located event of the Italian Conference on Computational Linguistics-CLiC-it. The EVALITA projects are organized along selected tasks, providing participants with opportunities to discuss and explore both emerging and traditional areas of Natural Language Processing and Speech for Italian. Indeed, many shared tasks, covering the analysis of both written and spoken language at various levels of processing, have been proposed within EVALITA since its first edition in 2007; NEEL-IT - Named Entity rEcognition and Linking in Italian Tweets, Named Entity Recognition on Transcribed Broadcast News, Entity Recognition and Named Entity Recognition. The proceedings of EVALITA projects are available as open access publications on the following platforms; CEUR open access platform, Academia University Press website,

Pisa University Press and Conference of the Italian Association for Artificial Intelligence amongst others.

3.2 Previous works on Named Entity Recognition

In Named Entity Recognition (NER) tasks, systems are required to recognize the Named Entities occurring in a text. In the first four editions of EVALITA, four types of entities are distinguished such as: Person, Organization, Location and Geo-Political Entities. The first three editions uses I-CAB data, the corpus of (written) news stories taken from local newspaper L'Adige. The fourth edition is based on spoken news broadcasts provided by the local broadcaster RTTR. The challenge has gradually developed to the trends and user behaviors in real-time. The most representative machine learning approaches used in NER are Hidden Markov Model (HMM), Maximum Entropy, Support Vector Machines (SVMs), and Conditional Random Fields (CRFs) (Zanoli, Pianta, and Giuliano, 2009).

The technologies used in the different approaches are:

- Support Vector Machines: Yashar Mehdad (Mehdad, Scurtu, and Stepanov, 2009) used YAMCHA, an implementation of SVMs, for feature extraction and selection. Rigo (Rigo, 2009) also uses a SVM classifier for machine learning and was trained on a large number of static and dynamic features; For development he used the open source chunker YamCha. Whereas the recognition of person names and geopolitical entities was satisfactory, the system performance according to locations and organizations was rather weak with an overall F1 measure of 74.98%.

- MaxEnt classifier. Attardi (Attardi et al., 2009) trained a Maximum Entropy tagger that is a generic, customizable text chunker. The tagger uses a variety of features, both local and global, which can be specified in a configuration file and employs dynamic programming to select accurate sequences of tags. The final accuracy was further improved by applying simple symbolic rules.

- CRF classifier. Riccardi (Nguyen, Moschitti, and Riccardi, 2009) uses a CRF as later described in this section.

- Other classifiers and modern approaches such as the Deep Learning approach were used as discussed below.

3.2.1 Bi-directional LSTM-CNNs-CRF for Italian Sequence Labeling

Neural networks have been widely used in natural language tasks because of their ability to learn features from complex tasks. In this paper, the author proposed a Deep Learning architecture for sequence labeling based on a state of the art model that exploited both word and character level representations through the combination of bidirectional LSTM, CNN and CRF. He evaluated the proposed method on three Natural Language Processing tasks for Italian: Named Entity Recognition, PoS-tagging of tweets and Super-Sense Tagging. In this paper for NER, he adopted the I-CAB dataset of 2009 EVALITA edition, containing 525 news for training and 180 for testing for a total number of 11,410 annotated entities for training and 4,966 ones for testing. The dataset is provided in the IOB2 format. He employed different configurations, the best result is achieved with *no-case-sensitive* configuration by applying lowercase of words for both word embeddings and the lookup table. In this result he achieved the best F1 result of 82%. The system outperformed the first three EVALITA 2009 participants who adopted classical classification methods as shown in **table1**. Results obtained show that the system is able to achieve state

of the art performance and in some cases exceeds the best systems previously developed for the Italian. This system achieved good results in all the tasks without incorporating any hand-crafted features. Analysis on result indicated that building word embeddings on appropriate corpora is very valuable. It was also observed that post-feature was very important in the SST task which is not able to generalize a good model without this feature (Basile, Semeraro, and Cassotti, 2009).

3.2.2 Named Entity Recognition through Redundancy Driven Classifiers

In the built system, Typhoon, two different classifiers (CRFpp and HMM) are combined in a cascade to exploit Data Redundancy and Patterns extracted from a large text corpus. In the first phase, CRF++ classifier, a customizable and open source implementation of Conditional Random Fields (CRFs) is trained on the training data and used to identify Named Entities then classify them; In the second phase, HMM classifier based on disambig, an implementation of the Hidden Markov Models was trained on the data produced by the previous phase, recognizing entities both in the training and test data. In this task, large feature selection played a key role wherein we used features such as; the word, unchanged case, lowercased, patterns, PoS, prefixes, suffixes, orthographic information and gazetteer. The system performed the best at EVALITA 2009, with an F1 of 0.82 as shown in Table 3.1. The two classifiers took advantages from each other to obtain high accuracy (Zanoli, Pianta, and Giuliano, 2009).

3.2.3 Bidirectional Sequence Classification for Named Entities Recognition

In this paper, the author presented a system for Named Entities Recognition, based on the Perceptron Algorithm. The system applied a more complex semi-supervised training approach, particularly extending the Guided Learning (GL) framework with moderate execution time, uncompromised learning capability and quality of the results. GL does not suffer from the label bias problem compared to other approaches. In the proposed framework, the order of the inference is not forced into a monotonic behavior (left-to-right), but is learned together with the parameters of the local classifier. The system tested on the task of Italian NER at EVALITA 2009 obtained the second position, with an F1 measure of 81.46%, further proved the validity of Guided Learning (Gesmundo, 2009).

3.2.4 Structural Reranking Models for Named Entity Recognition

In this paper, the author employs discriminative reranking technique and a combination of two state-of-the-art NER learning algorithms (CRFs and SVMs) for incorporating global features in named entity recognizers. Two kinds of features (flat and structured) are employed by the reranker . Whereas the flat features were generated by a polynomial kernel encoding entity, the structured tree kernels are used to model dependencies among candidate examples. This method first employs a basic probabilistic model (built with CRFs or SVMs) to generate a global hypotheses of NE annotation (top-N candidates) and then applied a reranker, a classifier based on SVMs and tree kernels, to select the best hypothesis. The experiment was employed on both Italian EVALITA 2009 and the English CoNLL 2003 datasets and it showed a large improvement on CRFs by 3.99% and 3.13% to values 84.33% and 87.99% respectively. Furthermore, the analysis revealed that both kernels provided improvement in comparison to the CRFs baseline and suggests synergy that their combination improves CRFs even much more than the total of the previous individual parts. The main contribution of this approach was the use of structural kernels for representing hypotheses of the NE annotation over an entire sentence. Another contribution

of this paper was the joint model, a combination of tree kernels with global features encoded in traditional vectors. The joint model was effective since it captured two different important aspects of the sentence annotation. The study further demonstrated that composite kernel is very effective for reranking named entity sequences. The evaluation on the English data reaches the state-of-the-art while in Italian corpus outperforms the best methods previously reported whereas on the English data it reaches the state-of-the-art (Nguyen and Moschitti, 2012).

3.2.5 Deep neural networks for named entity recognition

In this paper, Bodanita introduced a Deep Neural Network (DNN) for engineering Named Entity Recognizers (NERs) in Italian. To predict tags, their network used a sliding window of word contexts. It relied on a simplified word-level log-likelihood as a cost function and used a new recurrent feedback mechanism to realize that the dependencies between the output tags are appropriately modeled. They proposed RCWN (Recurrent Context Window Network) for modeling dependencies between labels adopted from CWN applied by Collobert to avoid its limitations. The models were evaluated on the Evalita 2009 dataset, applying 10fold crossvalidation. The model using both gazetteer and pretraining outperforms all the participating systems. It is previously observed that Nguyen et al. (2010) obtained better results using a CRF classifier followed by a reranker based on tree kernels (multiple learning approaches). However, this approach is simpler only using one learning algorithm. In fact this model outperformed Nguyen CRF baseline (input to the later tree-kernel based reranker). It can be transitively related that the model output can be used as input to the reranker with a likeliness to produce a further improvement over SOTA (Bonadiman, Severyn, and Moschitti, 2015).

3.2.6 Other techniques

Yashar Mehdad employs Support Vector Machines learning algorithmand feature extraction and selection. The system structure is composed of two main parts, Feature Extraction and Feature Selection. The former is based on the YAMCHA classifier machine. YAMCHA uses Support Vector Machine learning algorithm for classification. The latter subsequently, selected the set of features based on the experiments and results. The system performedthe at EVALITA 2009 with an overall F-measure of 81.09%, which has less than one percent gap with the best result(82%).According to the results, morphological features were helpful in improving the system (Mehdad, Scurtu, and Stepanov, 2009).

Stefan Rigo also used the SVM classifier for machine learning. In his approach, he trained on a large number of static and dynamic features including Gazetteers, orthographic and morphological information. YAMCHA was used for development. He obtained an overall F1 measure of 74.98% and took the fifth position of the seven EVALITA 2009 contestants. This system can easily be adapted to other languages (mainly by the use of different gazetteers) as It was conceived to be language independent (Rigo, 2009).

The Tanl system proposed by *Giuseppe Attardi* tagger is a generic, customizable text chunker that uses a Maximum Entropy classifier for learning how to chunk texts and can be applied to NLP tasks such as Named Entity recognition , POS tagging and Super Sense tagging. Maximum Entropy technique is more efficient than SVM and achieves similar levels of accuracy when complemented with dynamic programming. A remarkable aspect of Tanl is that it can do without POS features. The system obtained F1 measure of 73.16% (Attardi et al., 2009).

In their paper titled *Conditional Random Fields: Discriminative Training over Statistical features for Named Entity Recognition*, *guyen* implemented two learning algorithms for Named

Participant	FB1	Prec	Rec	GPE	LOC	ORG	PER
NguyenRR	84.33	85.99	82.73				
Bonadimanet	83.80	85.03	82.64				
UNIBA	82.34	82.86	81.82	85.61	62.20	65.87	92.39
ZanoliPianta	82.00	84.07	80.02	85.13	51.24	70.56	88.31
Gesmundo_r2	81.46	86.06	77.33	83.36	50.81	71.08	87.41
RGB_r2	81.09	83.20	79.08	85.25	52.24	69.61	86.69
RGB_r1	80.90	83.05	78.86	85.19	54.62	69.41	86.30
Nguyen_r1	79.77	82.26	77.43	82.85	42.34	67.89	86.44
Nguyen_r2	79.61	81.65	77.67	82.49	50.85	67.38	86.25
Gesmundo_r1	76.21	83.92	69.79	79.07	47.06	64.67	82.04
Rigo_r2	74.98	81.08	69.73	75.96	38.32	60.36	83.18
Rigo_r1	74.34	80.71	68.91	75.77	31.16	59.87	82.38

TABLE 3.1: Showing system results in terms of F-Measure, Precision and Recall (overall and for different types of Named Entities).

Entity Recognition with a large number of features; one making use of CRFs and the other SVMs. The two methods perform averagely the same F1 measure with model CRFs registering slightly better results (Nguyen, Moschitti, and Riccardi, 2009).

During the evaluation campaigns several tasks about NER have been organized such as; In 2007 (*Speranza, 2007*), Speranza, 2009, and **agnini2006annotazione2011** In several of the above papers we take into account the 2009 edition since the I-CAB dataset used in the evaluation is the same adopted in 2009. A different version of I-CAB was used in 2007, whereas in 2011 the task was focused on data transcribed by an ASR system. The I-CAB dataset entails a set of news annotated manually with four types of entities: GPE (geopolitical), LOC (location), ORG (organization) and PER (person). The dataset contains 525 news for training and 180 for testing for a total number of 11,410 annotated entities for training and 4,966 ones for testing with a higher percentage of PER Entities, followed by ORG and GPE Entities and a small number of LOC Entities. The dataset is provided in the IOB2 format (Speranza, 2009). Table4 reports evaluation results in terms of precision (P), recall (R) and F1-measure (F1) for overall system. F1-measure for the respective entity types are also presented in this table. The evaluation results correspond to different configurations of the system with respect to the participants Nguyen_RR(Nguyen, TrucVien, Moschitti) (Nguyen and Moschitti, 2012), (Bonadiman, Severyn, and Moschitti, 2015), UNIBA(Pierpaolo, Giovanni), ZanoliPianta (Zanoli, Pianta, and Giuliano, 2009). The results obtained by the top systems shown in Table 3.1 have F-measure ranging from 82.34% to 61.03% with top five systems scoring above 79%. The bestscoring system (i.e. UNIBA) achieved 82.34%, the second (i.e. ZanoliPianta) 82%, the third (i.e. Gesmundo) 81.46%, the forth (i.e. RGB) 81.09% and the fifth (Nguyen) 79.77% (see Table 4). Generally the top five systems obtained very closely ranged F1 scores. When the results are compared in terms of Precision and Recall, It is clear that all the systems obtained higher values for Precision than for Recall. The best scoring system in EVALITA 2009 was ZanoliPianta . Note that Nguyen et al. (2010) obtained best results, 84.33%, by using re-ranking techniques based on tree kernels and the combination of two state of the art NER learning algorithms: conditional random fields and support vector machines. However, Bonadimanet's approach used only one learning algorithm, a simpler model than models applying multiple learning approaches, such as those in (Nguyen and Moschitti, 2012) and (Zanoli, Pianta, and Giuliano, 2009). Thus it is likely that applying Nguyen's reranker on top of Bonadimanet's

model output might produce a further improvement over the state of the art Basile, Semeraro, and Cassotti, 2009, Nguyen and Moschitti, 2012 and Speranza, 2009.

Chapter 4

Experiment and Evaluation

In this chapter we explain the proposed work, Its implementation on a well known dataset and discuss the experimental results obtained.

4.1 Dataset

In this section we describe the datasets used in the implementation of this work, the preprocessing steps we performed on the data, the format of the data and the method that was used for system evaluation.

4.1.1 Dataset Description

The project uses the Italian Content Annotation Bank (I-CAB) dataset, a corpus of Italian news documents annotated using diverse kinds of semantic information. The same dataset was used as *EVALITA 2009* Italian dataset. The Autonomous Province of Trento has funded a three year project Ontotext leading to the creation of I-CAB. Ontotext is targeted on the study and development of innovative knowledge extraction approaches producing information with less or no noise. *Callisto*, a freely distributed tool that supports any Unicode-supported language and accepts files encoded as UTF8, US-ASCII and other character encoding, was used to annotate I-CAB (Magnini et al., 2006). The annotation of I-CAB was restricted to four semantic types as defined below;

- Person (PER): These entities include both single individuals and groups of people (including family names). Examples of entities type PER are; Laura, Carlo, Giovanni Paolo and Antonio.

- Organization (ORG): Agencies, corporations, and other groups of people defined who must be defined by a formally established association (organizational structure). Examples of entities of type ORG are; Government organizations (*Guardia di Finanza* in the sentence "L'auto e stata posta sotto sequestro dalla *Guardia di Finanza*"), Commercial organizations (*Stazione* in the sentence "Sono Rita e Tamara che gestiscono il bar *Stazione*"), Educational organizations (*Universita di Pisa*in the sentence "*Universita di Pisa* avra un nuovo rettore"), Entertainment organizations (*Rem* in the sentence "I *Rem* ORG presenteranno il loro nuovo album"), Non Governmental organizations (*Brigate Rosse* in the sentence "Nel muro di silenzio delle nuove *Brigate Rosse* si e finalmente aperto uno squarcio"), Media organizations (*Al Jazira* in sentence "Arrestato l'inviato di *Al Jazira*"), Medical science organizations (*CNR* in sentence "La rassegna e stata promossa dal *CNR*") and Sports organizations (*A1* in sentence "Ho colto al volo l'opportunità di giocare in *A1*")

- GeoPolitical Entity (GPE): Geographical regions defined by political and/or social groups (a nation, Its region, government, or people). Examples of entities of type

TABLE 4.1: Number of news stories per category of training data

	09/07	**09/08**	**10/07**	**10/08**	**Total**
News	23	25	18	21	87
Economy	13	15	12	15	54
Culture	20	18	16	18	72
Sport	29	41	27	26	123
Local	46	43	49	51	189
TOTAL	131	142	122	130	525

TABLE 4.2: Quantitative data about the training and test data.

	Training	**Test**
News stories	525	180
Sentences	11,227	4,136
Tokens	212,478	86,419
Tokens/news story	404.7	480.1

GPE are; Continents (*Asia*), Nations (*Italia, Stati Uniti*), States and provinces (*Florida, Toscana, Alto Aldige*), Counties and districts (*Canton Ticino*) and Population centers (*Trento*).

- Location (LOC): Geographical Entities such as geographical areas and landmasses, bodies of water, and geological formations. Examples of entities of type LOC are; Adresses (*Via Nazionale 12*), Celestial bodies (*Il pianeta Venere*), WaterBodies (Il *Po*, Il Mar *Mediterraneo*) and Natural land regions (Il Monte *Bondone*).

4.1.2 Data Format

The *CoNLL-2003* named entity data consists of eight files covering two languages: English and German (Tjong Kim Sang and De Meulder, 2003). The CoNLL English dataset, CoNLL 2003 was used. It is an English dataset created within the shared task of CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003). It is made up of a collection of news wire articles from the Reuters Corpus and annotated with four different entity types: Location (LOC), Person (PER), Organization (ORG) and Miscellaneous name (MISC). The training and the

TABLE 4.3: Quantitative data about the Named Entities in the training and in the test data.

	Training		**Test**	
GPE	2,813	(24.66%)	1,143	(23.02%)
LOC	362	(3.17%)	156	(3.14%)
ORG	3,658	(32.06%)	1,289	(25.96%)
PER	4,577	(40.11%)	2,378	(47.88%)
Total	11,410		4,966	

TABLE 4.4: Number of named entities per data file.

	Training	**Development**	**Test**
LOC	7140	1837	1668
MISC	3438	922	702
ORG	6321	1341	1661
PER	6600	1842	1617
Total	23499	5942	5738

TABLE 4.5: Number of articles, sentences and tokens in each data file.

	Training	**Development**	**Test**
Articles	946	216	231
Sentences	14,987	3,466	3,684
Tokens	203,621	51,362	46,435

development datasets are news feeds from August 1996, whereas the news feeds from December 1996 is contained in the test set. Subsequently, the test dataset named entities are considerably different from the ones that come from the training or the development set. Table 4.5 contains an overview of the sizes of the data files. The unannotated data contain 17 million tokens (English), whereas Table 4.4 having annotated data contains number of named entities per data file such as training, development or test.

Both development data and test data are consisting of two separate text files, with one token per line and an empty line after every sentence. The files contain one word per line with empty lines representing sentence boundaries. Furthermore, both development and test data have been annotated with Part of Speech information (PoS-tag) using the Elsnet tagset for Italian (It is available for download at; `http://evalita.itc.it/tasks/elsnet-tagset-IT.pdf`). Pre-processing for both the training data and the test data was provided for accessibility, i.e. sentence splitting and Part of Speech tagging (using the ELSNET tagset for Italian) (Speranza, 2009). Both development data and test data files consist of a number of columns separated by a blank. The columns respectively contain:

- the first column contains the *token*;
- the second column contains the *PoS-tag*;
- the third column contains the *Adige news story* to which the token belongs.

Note that only development(training) data are annotated with Named Entities in the IOB2 format. The fourth column contains this entity tag. Furthermore, the entity tag contains two parts namely:

- The IOB2 tag: *'B'* (for '**B**egin') denoting the first token of a Named Entity, *'I'* (for '**I**nside') is used for all other tokens in the Named Entity, and *'O'* (for '**O**utside') is used for all other words;
- The Named Entity type tag (only for tokens belonging to Named Entities): *PER* (for Person), *ORG* (for Organization), *GPE* (for Geo-Political Entity), or *LOC* (for Location).

TABLE 4.6: Extract from training data illustrating data format.

presidente	SS	adige20041008_id414178	O
della	ES	adige20041008_id414178	O
giunta	SS	adige20041008_id414178	B-ORG
provinciale	AS	adige20041008_id414178	I-ORG
Lorenzo	SPN	adige20041008_id414178	B-PER
Dellai	SPN	adige20041008_id414178	I-PER
,	XPW	adige20041008_id414178	O
in	E	adige20041008_id414178	O
in	E	adige20041008_id414178	O

TABLE 4.7: Input format (example available at: http://evalita.itc.it/tasks/ner-input-sample.txt).

il	RS	adige20041008_id414157
capitano	SS	adige20041008_id414157
della	ES	adige20041008_id414157
Gerolsteiner	SPN	adige20041008_id414157
Davide	SPN	adige20041008_id414157
Rebellin	SPN	adige20041008_id414157
ha	VIY	adige20041008_id414157
allungato	VSP	adige20041008_id414157

The examples in Table 4.6 illustrate development data format:

The system takes as input a three-column file (see Table 4.7) and produces as output a four-column file annotated with Named Entities in the IOB2 format (see Table 4.8).

4.2 Evaluation Metrics

Performance of our NE system is evaluated on individual named entities scores using three techniques: Precision, Recall and F-Measure as described in the EVALITA project (Speranza, 2009).

Precision indicates the percentage of correct positive predictions and is computed as the ratio of the number of Named Entities correctly identified by the system (True Positive) to the total number of Named Entities identified by the system (True Positive (TP) plus False Positive (FP)), as shown in Equation 4.1.

$$Precision = \frac{TP}{(TP + FP)} \tag{4.1}$$

Recall demonstrates the percentage of positive cases recognized by the system and is computed as the ratio of the number of NEs correctly identified by the system (True Positive (TP)) to the number of NEs that the system was expected to recognize (True Positive (TP) + False Negative (FN)), as shown in Equation 4.2.

$$Recall = \frac{TP}{(TP + FN)} \tag{4.2}$$

TABLE 4.8: Output format relative to input example of Table 4.8 (example available at: http://evalita.itc.it/tasks/ner-output-sample.txt).

il	RS	adige20041008_id414157	O
capitano	SS	adige20041008_id414157	O
della	ES	adige20041008_id414157	O
Gerolsteiner	SPN	adige20041008_id414157	B-ORG
Davide	SPN	adige20041008_id414157	B-PER
Rebellin	SPN	adige20041008_id414157	I-PER
ha	VIY	adige20041008_id414157	O
allungato	VSP	adige20041008_id414157	O

F-measure or balanced F-score (FB1 score) is the weighted harmonic mean of precision and recall as shown in Equation 4.3. At the EVALITA, CLiC-it, MUC-6 and MUC-7 conferences, systems were judged based on their F-measures. Consequently, this is the score which we will be using in this work to judge the quality of our named entity system.

$$FB1 = \frac{2(Precision * Recall)}{(Precision + Recall)} \tag{4.3}$$

In general, there is a trade-of between precision and recall. If the method outputs entities very conservatively, that is, reports only if it is absolutely certain of the entity, it can achieve very high precision but might probably suffer a loss in recall. Conversely, if the method outputs entities more aggressively, then it will obtain higher recall while losing precision. F1 measure, the harmonic mean of precision and recall, is a single umber that captures both precision and recall.

The output of *CRF++* is compatible to *CoNLL 2000* shared task. This therefore allows us to use the perl script *conlleval.pl* to evaluate our system outputs. The perl script gives us a list of F-measures with respect to our feature templates. The script also evaluates the result based on the accuracy measures; Global accuracy, precision and recall. However, for our evaluation interest we consider precision, recall and f-measure with our final evaluation focus on the f-measure. However, the perceptron algorithm output requires reformation in order for it to be evaluated by the same perl script. Therefore, preprocessing tools are applied to obtain the required format.

4.3 The baseline algorithm

As the baseline model we selected Conditional Random Fields (CRFs) . CRF is a probabilistic framework for labeling and segmenting structured data, such as sequences, trees. The underlying idea is that of defining a conditional probability distribution over label sequences given a particular observation sequence rather than independence assumptions required by HMMs. Otherwise, Hidden Markov models and other discriminative Markov models are prone to the label bias problem that is effectively solved by CRFs.

The named-entity recognition (NER) task is framed as a subtask of IE, involving identification of proper names in texts, and classifying them into a set of predefined categories of classes. As earlier hinted, we follow the IOB notation format where NE tags have the format B-TYPE, I-TYPE or O, which mean that the word is a beginning, a continuation or not part of an entity at all respectively. The examples below illustrate the use of the tags

with their corresponding words;
presidente/O della/O giunta/B-ORG provinciale/I-ORG Lorenzo/B-PER Dellai/I-PER ,/O in/O.

For our experiments, we used CRF++ to build our recognizer. CRF++, an open source and easily customizable implementation of CRFs for labeling sequential data, is also a model trained discriminatively with the unigram and bigram features. Centered on the target word w, features are extracted from a window size of n words. *W* is the one we desire to classify with the tagsets *B, O, I tags*. The features we employed are divided into two categories; Basic and combined features (()nguyen2012structural) as described in the subsection 4.3.1 below:

4.3.1 Features

The features are extracted from a window of k words centered in the target word w (i.e. the one we want to classify with the B, O, I tags). The features we used can be divided into 2 categories; Basic features and Combined features as described below (Nguyen and Moschitti, 2012):

Basic features: The basic features are directly derived from the word itself and is described as a small and primitive units as illustrated below:

- w_i is the i^{th} word, for $i = 1...n$
- l_i is the lower-case of the word
- $p1_i, p2_i, p3_i, p4_i$ are the four prefixes of w_i of order $1, 2, 3$ and 4 respectively.
- $s1_i, s2_i, s3_i, s4_i$ are the four suffixes of w_i of order $1, 2, 3$ and 4 respectively.
- f_i is the part-of-speech of w_i
- g_i is the orthographic feature that tests whether a word contains all upper case, initial upper case or all lower-case letters.
- k_i is a feature that tests whether a token is a word, a number, a symbol or a punctuation mark.
- o_i is the gazetteer feature. Here, we simply look up w_i in the prepared knowledge base.

Combined features: A Combined feature is a combination of two consecutive units that are used to derive the bigram features. In order to realize the maximum generalization, the subsequent units can be lexical words or any of their previously discussed basic features. An example is the text "king's palace" can have one feature as "king's/palace" and another feature as "JJ/NN" where JJ and NN are the part-of-speech of king's and palace. The features are illustrated below:

- The sequences $w_{i-2}/w_{i-1}, w_{i-1}/w_i$ and w_i/w_{i+1}
- Use l_i instead of w_i as in the above first point.
- Use $p1_i, p2_i, p3_i, p4_i$ instead of w_i as in the above first point.
- Use $s1_i, s2_i, s3_i, s4_i$ instead of w_i as in the above first point.
- Use f_i instead of w_i as in the first point.
- Use o_i instead of w_i as in the first point.

TABLE 4.9: Example of feature emplate for CoNLL 2000 shared task

```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]

U10:%x[-2,1]
U11:%x[-1,1]
U12:%x[0,1]
U13:%x[1,1]
U14:%x[2,1]
U15:%x[-2,1]/%x[-1,1]
U16:%x[-1,1]/%x[0,1]
U17:%x[0,1]/%x[1,1]
U18:%x[1,1]/%x[2,1]

U20:%x[-2,1]/%x[-1,1]/%x[0,1]
U21:%x[-1,1]/%x[0,1]/%x[1,1]
U22:%x[0,1]/%x[1,1]/%x[2,1]

# Bigram
B
```

- The sequences $w_{i-2}/f_{i-2}, w_{i-1}/f_{i-1}, w_i/f_i$ and w_i/w_{i+1}.
- The sequences $f_{i-2}/o_{i-2}, f_{i-1}/o_{i-1}, f_i/o_i$ and f_{i+1}/o_{i+1}.
- The sequences $w_{i-2}/o_{i-2}, w_{i-1}/o_{i-1}, w_i/o_i$ and w_{i+1}/o_{i+1}.

Gazetteer: The terms lexicon , list and dictionary are often used interchangeably with the term gazetteer. The gazetteers are built with names imported from different sources. Moreover, the gazetteer lists for Italian are extracted from *TextPro*, `http://textpro.fbk.eu/home`.

Feature templates: Since *CRF++* is designed as a general purpose tool, we specify the feature templates. The feature template file describes which features are used during training and testing. The templates correspond to the different features previously discussed such as; *Basic* and *Combined* features. Each line in the template file denotes one template. There are two types of templates specified with the first character of templates; *Unigram* template: first character, *'U'* and *Bigram* template: first character, *'B'* as described in the *URL* `https://taku910.github.io/crfpp/`. Table 4.9 demonstrates an example of feature emplate for CoNLL 2000 shared task. This template example shows that only combinations of previous output token and current token are used as bigram features.

TABLE 4.10: *Extended* feature set for the structured perceptron algorithm
y_i state at position i in the sequence, $i \in 1, ..., N$
x_i observation at position i in the sequence, $i \in 1, ..., N$
c_k particular state, $k \in 1, ..., N$
w_j particular state, $j \in 1, ..., J$ where J is the number of distinct observation labels

Condition	**Name**
$y_i = c_k$ & $i = 0$	Initial Features
$y_i = c_k$ & $y_{i-1} = c_l$	Transition Features
$y_i = c_k$ & $i = N$	Final Features
$x_i = w_j$ & $y_i = c_k$	Basic Emission Features
$x_i = w_j$ & w_j is uppercased & $y_i = c_k$	Uppercase Features
$x_i = w_j$ & w_j contains digit & $y_i = c_k$	Digit Features
$x_i = w_j$ & w_j contains hyphen & $y_i = c_k$	Hyphen Features
$x_i = w_j$ & $w_j[0...i]$ $\forall i \in [1, 2, 3]$ & $y_i = c_k$	Prefix Features
$x_i = w_j$ & $w_j[\|w_j\| - i..\|w_j\|]$ $\forall i \in [1, 2, 3]$ & $y_i = c_k$	Suffix Features

TABLE 4.11: *ID* feature set for the structured perceptron algorithm
y_i state at position i in the sequence, $i \in 1, ..., N$
x_i observation at position i in the sequence, $i \in 1, ..., N$
c_k particular state, $k \in 1, ..., N$
w_j particular state, $j \in 1, ..., J$ where J is the number of distinct observation labels

Condition	**Name**
$y_i = c_k$ & $i = 0$	Initial Features
$y_i = c_k$ & $y_{i-1} = c_l$	Transition Features
$y_i = c_k$ & $i = N$	Final Features

4.4 Structured Perceptron

We performed our experiment in parallel with the structured Perceptron. This is helpful in comparing the performances of the two algorithms. The structured perceptron (Collins, 2002) also described as the averaged version is a very simpllified algorithm relying on Viterbi decoding and very simple additive updates. In practice this algorithm behaves remarkably well in a variety of tasks yet very easy to implement making structured perceptron algorithm a natural first choice to try and test a new problem or a new feature set. The features used in this algorithm are *ID* feature set and *Extended* feature set. The *Extended* features are illustrated in Table 4.10. *ID* feature set in Table 4.11 replicates the features used by the *Hidden Markov Model* (*HMM*). *Extended* feature set contain additional features to those of the *HMM*, where they could not be included.

4.5 Feature Selection and Feature Utility Analysis

Feature selection plays a very crucial role in machine learning and NLP tasks. In our systems, we simply use features chosen from *Basic* and *Combined* features earlier discussed

in section 4.3.1. Experiments were then carried out for CRFs with different feature combinations where as those for the structured perceptron were predefined as *ID* Features in Table 4.11 and *Extended* features in Table 4.10. It turned out for CRFs that the *combined* features achieved the best results whereas structured perceptron realizes better results with *Extended* features.

The appropriate choice of features is essential for obtaining a good system for recognizing named entities. To find an optimal set of features, we have to check all possible combinations of them. But considering the excessively many resulting combinations, in this project we have opted for a selected combination of these features. In *CRFs* we start out with a very basic set of features. We perform these considering the feature templates of *CRF++*; *Unigram* and *Bigram* Table 4.9. Whereas in structured perceptron, we evaluated separately. A feature which results in the highest improvement is then preserved in the new feature set.

4.6 Experiments

Table 4.12 illustrates a sequence of experiments which led to the observed results achieved with the accompanying features used.

In the first experiment with Experiment ID $= 1$, I use CRF++, a baseline model to compare performances of English and ICAB datasets. I trained the CRF classifier on the entire training set (11,227 sentences in the Italian and 14,987 sentences in the English corpus) using a template file exploiting unigram and bigram features word and POS tag being used. Table 4.13 shows the performance of this experiment on the Italian and English test sets respectively w.r.t CRF.

In the second experiments with Experiment ID $= 2$, the task is to train crf++ using unigram and bigram features exploiting feature template mechanisms such as: bigram features(Basic feature), combined and bigram features (combined bigram features) and Basic bigram feature template. In the experiment I compare performances of Eng and ICAB datasets across different features (w_i, f_i). I trained the classifier with 11,227 sentences I-CAB-corpus and 14,987 sentences eng-corpus) using the feature template files prior described. These experiments did not involve other features such as; prefixes and suffixes. The results for the English and Italian sets are in Table 4.14 and Tables 4.15 respectively.

In the third experiments with Experiment ID $= 3$, the task is to train crf++ using bigram features exploiting *POS*, *prefixes* (P_1, P_2, P_3, P_4), *suffixes* (S_1, S_2, S_3, S_4) and *lower cases* with feature template mechanisms such as; Basic bigram features $(w_i, f_i, p_i, s_i, l_i)$ and combined bigram features. I trained the classifier with 11,227 sentences I-CAB-corpus and 14,987 sentences eng-corpus) using the described feature templates. Results are shown in Table 4.16 and Table 4.17.

The fourth Experiment with ID $= 4$ involves using bigram templates exploiting additional features such as; orthographic feature that tests cases and feature that tests whether a token is word, number, symbol, punctuation. We trained the classifier with 11,227 sentences I-CAB-corpus and 14,987 sentences eng-corpus exploiting basic and combined bigram feature templates. The results are in Table 4.18 and Table 4.19.

Experiment ID $= 5$ involves engaging an additional feature (gazetteer). This test was done only with the I-cab dataset since the gazetteer was from extracted the Italian knowledge base. This experiment all the discussed features. Similarly the experiment was with 11,227 sentences I-CAB-corpus exploiting the combined bigram feature templates. The result is reflected in Table 4.20.

TABLE 4.12: Description of experiments

Experiment ID	Features Used
1	Basic bigram features(w_i, f_i)
2	Combined bigram features (w_i, f_i)
3	Bigram features (w_i, f_i, p_i, s_i, l_i)
4	w_i, f_i, p_i, s_i, l_i, g_i, k_i
5	w_i, f_i, p_i, s_i, l_i, g_i, k_i, o_i
6	*Extended* features

The sixth Experiment with $ID = 6$ uses the *Extended* feature set for the structured perceptron algorithm discussed in subsection 4.4. In this experiment we customized the *Lisbon Machine Learning Summer School Lab Guide* (*LxMLS - Lab Guide*) (QUINTÃO, 2014) and the code contained in *Github account* `https://github.com/LxMLS/lxmls-guide`. We cloned this code which is freely available for public use. Both datasets were reformulated to be used as input to this package. We used python script to carry out this task such that the dataset assumed the structure of that used in the *LxMLS - Lab Guide*. Then we trained the perceptron using different feature sets for NER. In order to implement the structured perceptron algorithm, we edited file sequences/structured perceptron.py and implemented the function perceptron_update. The perceptron_update function should apply one round of the perceptron algorithm, updating the weights for a given sequence, and returning the number of predicted labels and the number of mistaken labels. Then, we adapted the function gradient_update defined in file sequences/structured_perceptron.py. We replaced the computation of posterior marginals by the Viterbi algorithm, and changed the parameter updates according to the described perceptron Algorithm. We provided 20 epochs, after training is done, we evaluated the learned model on the training, development and test sets as illustrated in Table 4.21. All experiments were run with epochs = 10 and using extended feature mapper. The accuracy plots on test sets across the different languages show a closely similar pattern as illustrated in the Figures 4.1, 4.2 and 4.3.

4.7 Results

Table 4.13 shows the performance on the Italian and English test sets respectively w.r.t CRF. This result in comparison with those reported in the paper (Nguyen and Moschitti, 2012) is: conll (ENG) 82.34% (-2.52) 84.86%, evalIta (ITA) 61.06% (-19.28) 80.34%. We have - 19.28% on the evalIta dataset versus -2.52% on Conll. It performs better with English set. Moreover, the NE classes, GPE and PER, achieve a rather good F1, while the recognition of ORG and LOC seems more problematic. This is in line with previous work according to which ORG seems to be the most difficult category to be learned. This demonstrates that the results on the Italian datasets are better, but still too low if we compare them with respect to the state of the art. Without necessarily exploiting complex features we can realize moderate accuracy once we incorporate the bigram features. Furthermore, the template file and other parameters could be better tuned to realize higher accuracy for the CRF++ model.

Tables 4.14 and 4.14 show performances on eng and i-cab data sets respectively Experiment ID = 2. We observe the following: Performance is highest using *combined and bigram features*; *combined features* generally contributes to high accuracy for both language; and *bigram features* seem to contribute less compared with the *combined features*. These results by

ICAB test.	precision	recall	$F_{\beta=1}$
LOC	79.41%	17.31%	28.42
GPE	79.46%	59.23%	67.87
ORG	63.74%	42.28%	50.84
PER	75.00%	56.64%	64.54
overall	73.40%	52.28%	61.06

Eng test	precision	recall	$F_{\beta=1}$
LOC	84.99%	83568%	84.27
MISC	86.61%	68.76%	76.66
ORG	81.71%	75.62%	78.54
PER	85.34%	85.94%	85.64
overall	84.59%	80.21%	82.34

TABLE 4.13: CRF++ results on the ICAB and English test set.

exploiting the combined features without necessarily incorporating other features obtains better accuracies than in Experiment with ID = 1.

Tables 4.16 and 4.17 show performances on eng and i-cab data sets respectively for Experiment ID $= 3$. We observe that Performance is highest using *Basic features* $+f_i -$ *combined* and *combined features* generally contributes to high accuracy for both languages. While we are quite close to the SOTA accuracy with icab dataset, we met the desired accuracy with the English set. The new features added such as prefixes, suffixes and lower cases strongly contribute to the increase in accuracy. Exploiting more features is key for further remarks.

Experiment ID $= 4$ results showing performances on eng and i-cab data sets are in Tables 4.18 and 4.19 respectively. Performance is highest using *combined features*; *combined features* generally contributes to high accuracy for both languages; There is negligible improvement compared to the accuracy of Experiment ID $= 3$. It is vital to carry out investigation with the gazetteer features.

Experiment ID $= 5$ involved use of all the eight different features described in subsection 4.3.1. The results in Table 4.20 shows that gazetteer feature plays a a very vital role in contributing to the accuracy. This feature, extracted from *TexPro* was relatively rich with the respective named entities. The relatively simplified *CRF++* model result is comparative with the state of the art results.

The perceptron algorithm run obtained the results summarized in Table 4.21. Using a similar feature set a *CRF* yields better results than *HMM*. The results vary across the different languages illustrating that *N.E.* systems may have to be rewritten whenever there is a shift in domain or language. Generally the system performs better in the English domain compared to the other languages. In addition,*N.E GPE* performs best for the Italian dataset with least performance in *ORG*. Figure 4.1. Observe the precision recall curves in Figure 4.1, Figure 4.2 and Figure 4.3. There is a level of observable noise in the dataset. The features adoptable for this task is so few and may require involving more features to have a better graph patter. However, there is a behavior of the curves are closely related to the expected pattern.

combined bigram.	precision	recall	$F_{\beta=1}$
LOC	86.23%	82.14%	84.14
MISC	91.48%	75.70%	82.85
ORG	83.74%	74.50%	78.85
PER	85.44%	85.99%	85.71
overall	86.15%	80.61%	83.29

Basic bigram	precision	recall	$F_{\beta=1}$
LOC	83.88%	84.10%	83.99
MISC	89.63%	67.46%	76.98
ORG	82.56%	75.17%	78.69
PER	84.62%	86.05%	85.33
overall	84.55%	80.11%	82.27

combined bigram.	precision	recall	$F_{\beta=1}$
LOC	85.37%	84.81%	85.09
MISC	90.50%	73.32%	81.01
ORG	82.68%	75.47%	78.91
PER	85.58%	86.64%	86.11
overall	85.53%	81.49%	83.46

Basic bigram.	precision	recall	$F_{\beta=1}$
LOC	84.97%	82.20%	83.56
MISC	90.15%	70.50%	79.12
ORG	83.91%	71.96%	77.48
PER	85.23%	84.91%	85.07
overall	85.52%	78.91%	82.08

TABLE 4.14: CRF++ results on the eng test set w.r.t. feature templates.

combined bigram.	precision	recall	$F_{\beta=1}$
GPE	78.34%	65.18%	71.16
LOC	74.42%	20.51%	32.16
ORG	70.01%	48.72%	57.46
PER	80.77%	58.28%	67.71
overall	77.38%	56.20%	65.11

Basic bigram	precision	recall	$F_{\beta=1}$
GPE	78.55%	61.50%	68.99
LOC	85.19%	14.74%	25.14
ORG	71.69%	42.82%	53.62
PER	77.85%	55.42%	64.75
overall	76.69%	52.28%	62.17

combined bigram.	precision	recall	$F_{\beta=1}$
GPE	79.10%	64.92%	71.31
LOC	83.33%	19.23%	31.25
ORG	71.63%	47.01%	56.77
PER	79.31%	57.70%	66.80
overall	77.46%	55.38%	64.58

Basic bigram.	precision	recall	$F_{\beta=1}$
GPE	78.01%	63.95%	70.29
LOC	73.68%	17.95%	28.87
ORG	69.49%	44.53%	54.28
PER	80.79%	56.60%	66.57
overall	77.27%	53.95%	63.54

TABLE 4.15: CRF++ results on the ICAB test set w.r.t. feature templates.

Basic.	precision	recall	$F_{\beta=1}$
LOC	90.93%	92.22%	91.57
MISC	89.64%	81.67%	85.47
ORG	86.59%	80.39%	83.37
PER	90.05%	91.37%	90.70
overall	89.53%	87.65%	88.58

Combined.	precision	recall	$F_{\beta=1}$
LOC	92.38%	92.38%	92.38
MISC	90.05%	82.43%	86.07
ORG	85.69%	80.84%	83.19
PER	91.83%	90.93%	91.38
overall	90.40%	87.78%	89.07

TABLE 4.16: CRF++ results on eng test set with respect to feature templates.

Basic.	precision	recall	$F_{\beta=1}$
GPE	80.23%	73.14%	76.52
LOC	75.00%	28.85%	41.67
ORG	69.13%	60.12%	64.32
PER	84.58%	75.90%	80.01
overall	79.44%	66.69%	74.25

Combined.	precision	recall	$F_{\beta=1}$
GPE	78.70%	72.09%	75.25
LOC	77.46%	35.26%	48.46
ORG	67.55%	61.06%	64.14
PER	85.33%	76.32%	80.58
overall	78.93%	70.10%	74.25

TABLE 4.17: CRF++ results on ICAB test set with respect to feature templates.

Basic.	precision	recall	$F_{\beta=1}$
LOC	90.93%	92.22%	91.57
MISC	89.64%	81.67%	85.47
ORG	86.59%	80.39%	83.37
PER	90.05%	91.37%	90.70
overall	89.53%	87.65%	88.58

Combined.	precision	recall	$F_{\beta=1}$
LOC	92.38%	92.38%	92.38
MISC	90.05%	82.43%	86.07
ORG	85.69%	80.84%	83.19
PER	91.83%	90.93%	91.38
overall	90.40%	87.78%	89.07

TABLE 4.18: CRF++ results on eng test set with respect to feature templates.

Basic.	precision	recall	$F_{\beta=1}$
GPE	79.02%	74.45%	76.67
LOC	73.33%	28.21%	40.74
ORG	67.68%	61.91%	64.67
PER	84.37%	78.09%	81.11
overall	78.59%	71.49%	74.87

Combined.	precision	recall	$F_{\beta=1}$
GPE	79.70%	73.84%	76.66
LOC	78.95%	28.85%	42.25
ORG	68.40%	62.14%	65.12
PER	83.42%	78.30%	80.78
overall	78.60%	71.53%	74.90

TABLE 4.19: CRF++ results on ICAB test set with respect to the feature templates (no gazetteer).

gaz=1/0/.	precision	recall	$F_{\beta=1}$
GPE	87.75%	85.21%	86.46
LOC	93.33%	35.90%	51.85
ORG	80.27%	82.70%	81.47
PER	86.98%	85.95%	86.46
overall	85.40%	83.37%	84.37

TABLE 4.20: CRF++ results with gazetteers(1_s & 0_s).

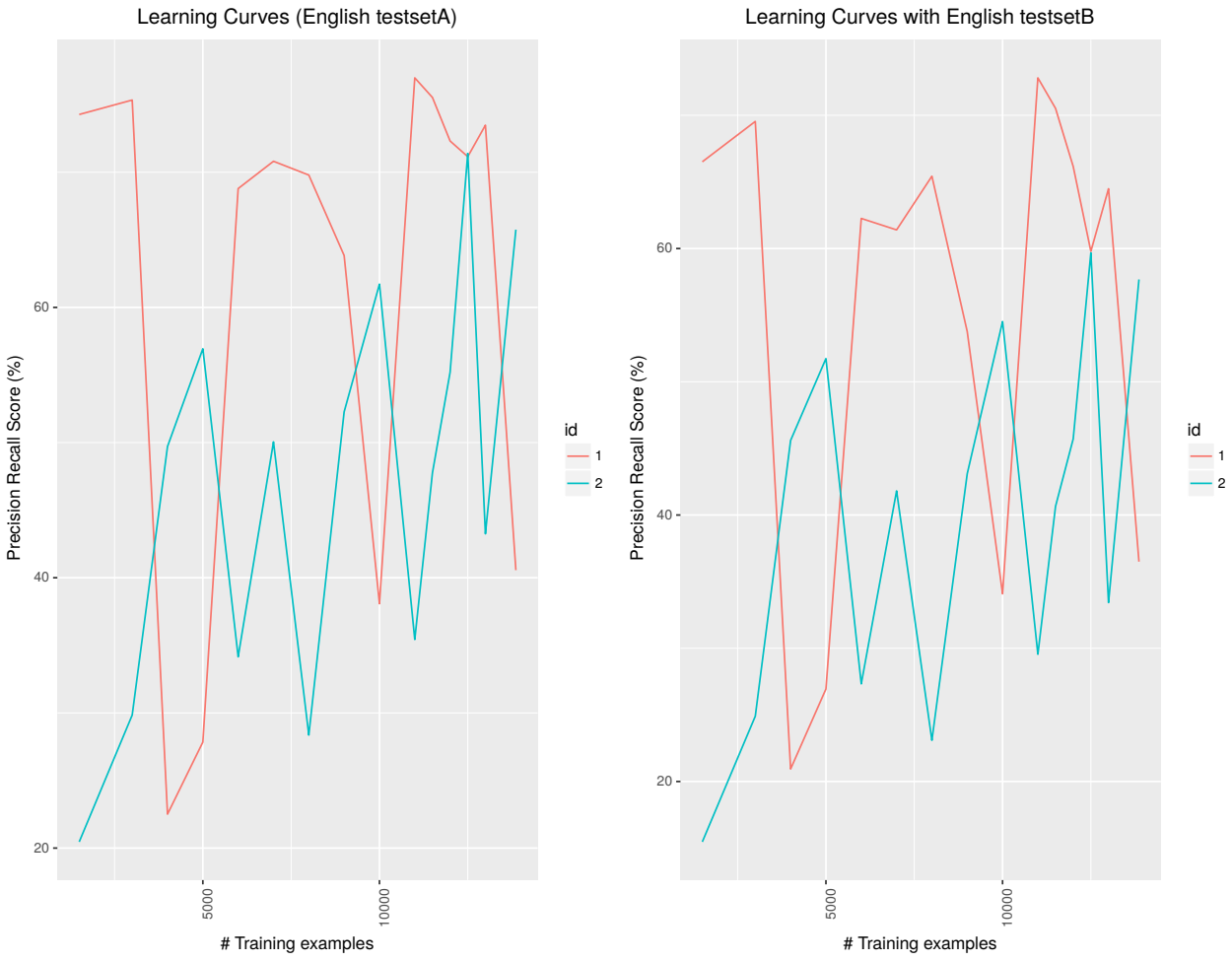

FIGURE 4.1: Precision Recall learning curves with respect to the English dataset

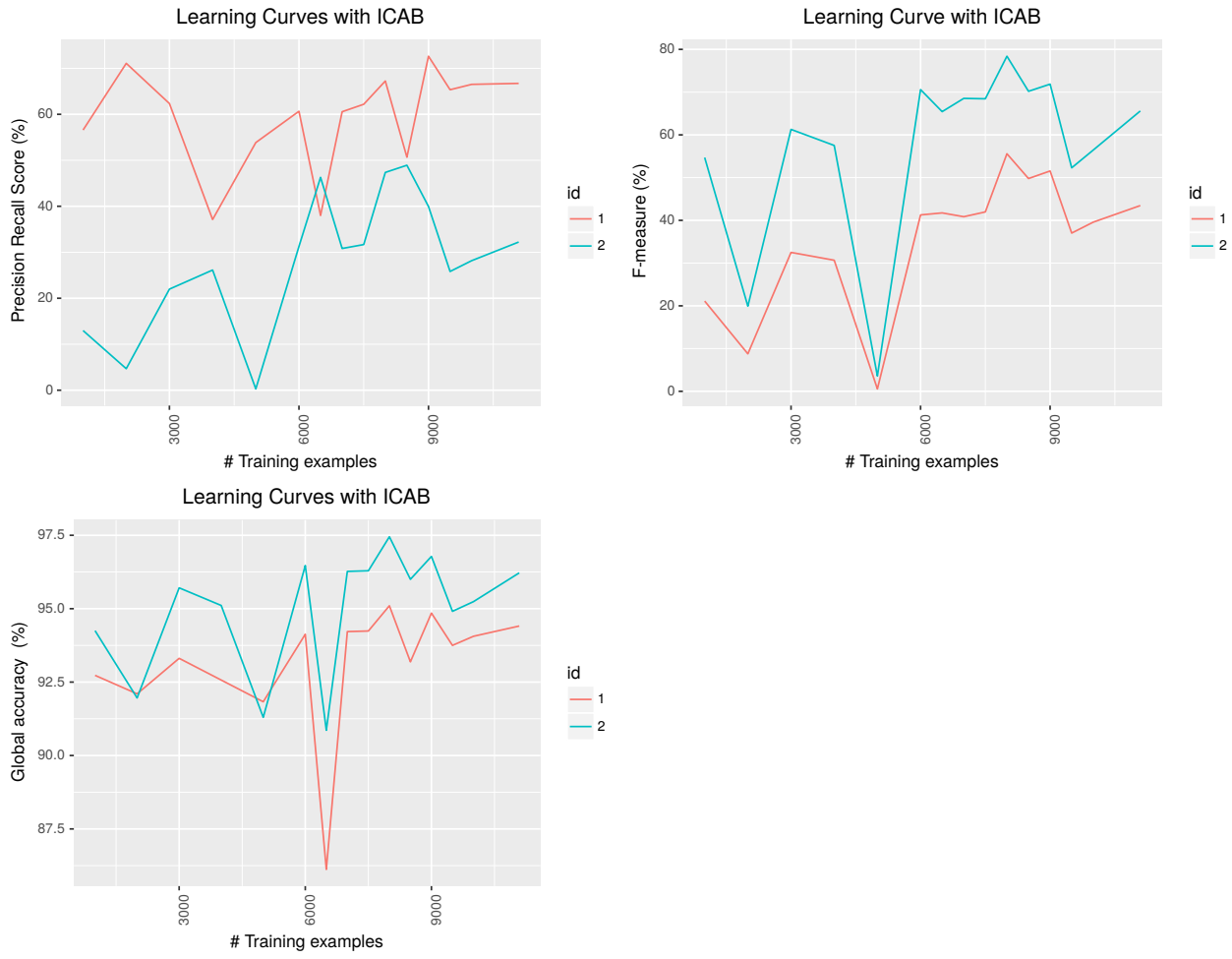

FIGURE 4.2: Precision Recall and Global accuracy with respect to the Italian ICAB dataset)

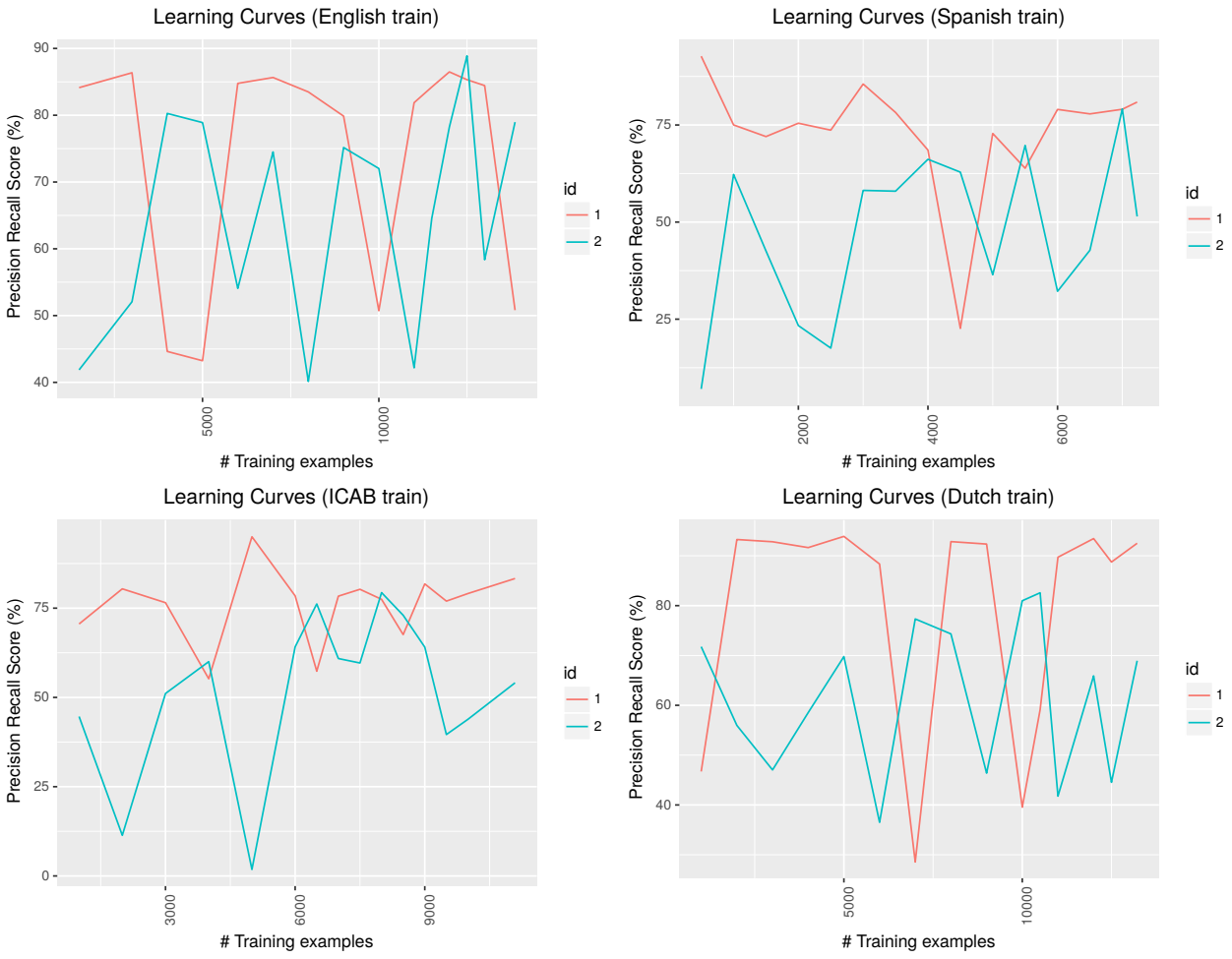

FIGURE 4.3: Accuracies on the training sets (i.e. global accuracy, precision, recall and f-measure) for the English, Spanish, Dutch and ICAB datasets.

4.7.1 System Analysis

In the previous section we identified the best configuration for the CRF and Perceptron models. We found out that Perceptron reaches its peak performance using *Extended* features of Table 4.10. Here, we ignore introducing results for the structured perceptron since only a few features were used. On the other hand, *CRF++* performs best when bigrams of basic and combined features discussed in subsection 4.3.1 is used. We also observe that introducing the gazetteer feature hugely contributed to escalating the performance of the *CRF++* model. Since the the gazetteer used is domain specific to Italian language, the feature was not used for the English data set. Therefore, the best English set was obtained from the bigrams of basic and combined features discussed in subsection 4.3.1 without using the gazeteer. Table 4.22, Table 4.23, Table 4.24 and Table 4.25 show a summary of the best results for the various categories. Table 4.22 and Table 4.23illustrate the performance of CRF++ on I-CAB and CoNLL English (CoNLL 2003) dataset s described in subsections 4.1.1 and 4.1.2 respectively. CRF++ performs best with named entities *PER* and *GPE* with respect to the 4.1.1. On the other hand, best named entities for (CoNLL 2003) are *PER* and *LOC* as this does not have named entity *GPE*. If we compare the results in terms of Precision and Recall, we can see that all the EVALITA systems in Table 3.1 obtained higher values for Precision than for Recall in all submitted runs. Similarly, our results report same pattern of behavior. Generally the results have values for F-Measure ranging from 84.33% to 61.03%. Most systems have obtained values of F-Measure between 63 and 69.

Participant	FB1	Prec	Rec	GPE	LOC	ORG	PER
NguyenRR	84.33	85.99	82.73				
Bonadimanet	83.80	85.03	82.64				
UNIBA	82.34	82.86	81.82	85.61	62.20	65.87	92.39
ZanoliPianta	82.00	84.07	80.02	85.13	51.24	70.56	88.31
Gesmundo_r2	81.46	86.06	77.33	83.36	50.81	71.08	87.41
RGB_r2	81.09	83.20	79.08	85.25	52.24	69.61	86.69
RGB_r1	80.90	83.05	78.86	85.19	54.62	69.41	86.30
Nguyen_r1	79.77	82.26	77.43	82.85	42.34	67.89	86.44
Nguyen_r2	79.61	81.65	77.67	82.49	50.85	67.38	86.25
Gesmundo_r1	76.21	83.92	69.79	79.07	47.06	64.67	82.04
Rigo_r2	74.98	81.08	69.73	75.96	38.32	60.36	83.18
Rigo_r1	74.34	80.71	68.91	75.77	31.16	59.87	82.38

ICAB test.	precision	recall	$F_{\beta=1}$
LOC	76.79%	27.56%	40.57
GPE	65.99%	65.99%	65.99
ORG	61.39%	24.12%	34.64
PER	42.75%	55.59%	48.33
overall	50.67%	48.92%	49.78

ICAB train.	precision	recall	$F_{\beta=1}$
LOC	80.73%	55.16%	65.54
GPE	70.66%	90.07%	79.19
ORG	72.65%	41.29%	52.65
PER	63.81%	89.47%	74.49
overall	67.59%	73.01%	70.19

Eng devel.	precision	recall	$F_{\beta=1}$
LOC	54.23%	55.91%	55.06
MISC	80.48%	68.4%	73.97
ORG	62.64%	44.59%	52.09
PER	68.94%	46.15%	55.28
overall	63.84%	52.27%	57.48

Eng test	precision	recall	$F_{\beta=1}$
LOC	47.62%	56.48%	51.67
MISC	70.18%	56.98%	62.89
ORG	59.72%	32.95%	42.47
PER	51.28%	51.28%	33.66
overall	53.77%	43.10%	47.85

Spanish dev.	precision	recall	$F_{\beta=1}$
LOC	64.75%	61.73%	63.20
MISC	43.57%	33.56%	37.91
ORG	62.06%	47.13%	53.57
PER	65.55%	44.84%	53.26
overall	61.80%	48.41%	54.29

Spanish test	precision	recall	$F_{\beta=1}$
LOC	68.84%	52.58%	59.62
MISC	42.03%	36.47%	39.06
ORG	68.65%	64.29%	66.40
PER	73.92%	65.17%	69.27
overall	67.26%	58.25%	62.43

Dutch devel.	precision	recall	$F_{\beta=1}$
LOC	78.88%	27.49%	40.77
MISC	66.75%	35.97%	46.75
ORG	80.37%	31.96%	45.73
PER	76.17%	27.90%	40.84
overall	74.28%	31.15%	43.89

Dutch test	precision	recall	$F_{\beta=1}$
LOC	88.51%	38.31%	53.47
MISC	66.61%	37.00%	47.57
ORG	74.53%	36.46%	48.96
PER	76.97%	37.57%	50.49
overall	75.04%	37.29%	49.82

Eng(Basic + Combined features)	precision	recall	$F_{\beta=1}$
LOC	92.38%	92.38%	92.38
MISC	90.05%	82.43%	86.07
ORG	85.69%	80.84%	83.19
PER	91.83%	90.93%	91.38
overall	90.40%	87.78%	89.07

TABLE 4.22: CRF++ results on Eng).

ICAB(Basic + Combined features)	precision	recall	$F_{\beta=1}$
GPE	79.70%	73.84%	76.66
LOC	78.95%	28.85%	42.25
ORG	68.40%	62.14%	65.12
PER	83.42%	78.30%	80.78
overall	78.60%	71.53%	74.90

TABLE 4.23: CRF++ results on ICAB test set with respect to the feature templates (All_Basic + All_Combined features - gazetteer).

gaz=1/0/.	precision	recall	$F_{\beta=1}$
GPE	87.38%	86.00%	86.68
LOC	93.85%	39.10%	55.20
ORG	80.13%	83.24%	81.66
PER	87.64%	85.58%	86.60
overall	85.59%	83.61%	84.59

TABLE 4.24: Best Italian I-CAB CRF++ results with all features including gazetteers(All_Basic + All_Combined features + gazetteer).

gaz=1/0/.	precision	recall	$F_{\beta=1}$
GPE	87.75%	85.21%	86.46
LOC	93.33%	35.90%	51.85
ORG	80.27%	82.70%	81.47
PER	86.98%	85.95%	86.46
overall	85.40%	83.37%	84.37

TABLE 4.25: Best Italian I-CAB CRF++ results with all features including gazetteers(All_Basic + All_Combined features + gazetteer).

4.8 Conclusion

The system described in this report was conceived to be language independent and can thus can be easily adapted to other languages (mainly by the use of different gazetteers). According to the results, it can be argued that the location and organization gazetteers are not performing satisfactorily, although they contain most of the tokens belonging to those classes, which are found in the test set. A possible improvement could be obtained by the collection of predictive words, which could be implemented as a rule for "gazetteer disambiguation" for words found in more than one gazetteer.

In this paper we described our system developed for Named Entity Recognition task (NER) with features similar to those of (Nguyen and Moschitti, 2012). With these feature sets, using CRF++ model, we obtained very good results for Italian NER. Our accuracy is very high because of optimal exploitation of CRF++ feature templates. Gazetteer features were also very helpful in improving the system results. We also found out that the Named Entity classes PER and GPE reach quite good F1 values, while the recognition of LOC and ORG seemed problematic. This is in line with previous EVALITA and SOTA results in which ORG seems to be the most difficult to learn for Italian NE. Lack of resources (Gazetteer for LOC is least) might have caused the low LOC accuracy. The system described in this report was realized to be language independent and can thus easily be adapted to other domains/languages (especially by using gazetteers for the respective language domain). According to the results, it can also be argued that the location and organization gazetteers are not performing satisfactorily enough, although they contain most of the tokens belonging to those classes found in the test set. A possible improvement could be obtained by improving on the gazetteer lists and "gazetteer disambiguation for words found in more than one gazetteer".

CRF++ model performs well with a simplified feature template mechanism that makes very easy to integrate new unigram/bigram features. Finally, we embedded the CRF++ classifier in a UIMA Annotator so that it can work with the rest of the UIMA framework. Therefore, the tagger is freely available on Github as an Apache UIMA component.

In summary, we developed a state-of-the-art NER for Italian language and then released a UIMA annotator that wraps the Italian NER. In future, the output of CRF can be combined with a reranker to further improve its performance.

Bibliography

Attardi, Giuseppe et al. (2009). "The Tanl Named Entity Recognizer at Evalita 2009". In: *EVALITA*. Conference of the Italian Association for Artificial Intelligence.

Basile, Pierpaolo, Giovanni Semeraro, and Pierluigi Cassotti (2009). "Bi-directional LSTM-CNNs-CRF for Italian Sequence Labeling". In: *Proc. of Evalita*.

Bonadiman, Daniele, Aliaksei Severyn, and Alessandro Moschitti (2015). "Deep neural networks for named entity recognition in Italian". In: *CLiC it*, p. 51.

Borthwick, Andrew and Ralph Grishman (1999). "A maximum entropy approach to named entity recognition". PhD thesis. Citeseer.

Chinchor, Nancy and Patricia Robinson (1997). "MUC-7 named entity task definition". In: *Proceedings of the 7th Conference on Message Understanding*. Vol. 29.

Collins, Michael (2002). "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms". In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pp. 1–8.

Freund, Yoav and Robert E Schapire (1999). "Large margin classification using the perceptron algorithm". In: *Machine learning* 37.3, pp. 277–296.

Gesmundo, Andrea (2009). "Bidirectional sequence classification for named entities recognition". In: *Proceedings of Evaluation of NLP and Speech Tools for Italian*.

Kudo, Taku (2005). "CRF++: Yet another CRF toolkit". In: *Software available at http://crfpp. sourceforge. net*, p. 130.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In:

Magnini, Bernardo et al. (2006). "Annotazione di contenuti concettuali in un corpus italiano: I-CAB". In: *Proc. of SILFI 2006*.

Mehdad, Yashar, Vitalie Scurtu, and Evgeny Stepanov (2009). "Italian named entity recognizer participation in NER task@ Evalita 09". In: *Proceedings of EVALITA*.

Nguyen, Truc-Vien T and Alessandro Moschitti (2012). "Structural reranking models for named entity recognition". In: *Intelligenza Artificiale* 6.2, pp. 177–190.

Nguyen, Truc-Vien T, Alessandro Moschitti, and Giuseppe Riccardi (2009). "Conditional random fields: Discriminative training over statistical features for named entity recognition". In: *Proceedings of EVALITA 2009 workshop, the 11st International Conference of the Italian Association for Artificial Intelligence (AI* IA), Reggio Emilia, Italy*.

QUINTÃO, ME (2014). "Quotation attribution for portuguese news corpora". PhD thesis. Master's thesis, Técnico Lisboa/UTL, Portugal.

Rigo, Stefan (2009). "Named Entity Recognition for Italian using SVM". In:

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Speranza, Manuela (2009). "The named entity recognition task at evalita 2009". In: *Proceedings of the Workshop Evalita*.

Tjong Kim Sang, Erik F and Fien De Meulder (2003). "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition". In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pp. 142–147.

Zanoli, Roberto, Emanuele Pianta, and Claudio Giuliano (2009). "Named entity recognition through redundancy driven classifiers". In: *Proc. of Evalita* 9.