UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Automatic Data Recognition System in Natural Language

Presentado por Malte Jansen
en Universidad de Burgos — April 27, 2019
Tutor: Bruno Baruque Zanón

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Nombre del alumno, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, April 27, 2019

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Named Entity Recognition (NER) is a subtask of Natural Language Processing (NLP). It focuses on extracting and classifying named entities from text. The kind of texts that will be focused on are informal texts such as tweets and reddit posts. Different existing NER tools will be evaluated on a number of datasets in order to select the best approach and configuration for informal texts.

Keywords

keywords separated by commas.

Contents

List of Figures

List of Tables

Introduction

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetives

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Theoretical concepts

This section will contain descriptions of fundamental concepts applied to the project.

3.1 Natural Language Processing(NLP)

Natural language processing is is a field of computer science and plays a big part in it's sub-field of artificial intelligence.

Some of today's use cases are information extraction, machine translation, summarization, search and human-computer interfaces.[1]

It describes techniques in which natural language is processed into a formal representation which computer can understand so they can further work with the data. In order to do that the spoken or written human language has to be analysed. It is not enough for the computer to know the significance of each word or even each sentence. Otherwise a simple dictionary could be easily used. In order to understand a text the capture of complete text correlations is necessary. This is a challenge for computer because of the complexity of human language and it's ambiguity. Computer unlike humans can't use their experience to understand a text but rather have to use artificial language algorithms and techniques. One of those techniques is called Named Entity Recognition(NER).

[1] Cf. Jason Collobert Ronan; Westan. *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning.* NEC Labs America.

3.2 Named Entity Recognition(NER)

Named entity recognition refers to the extraction of important information from a text. In named entity recognition the task is to identify which information is important and to then categorise this information. This results in named entities, that are particular terms in a text that are more informative than others or have a unique context. It can be used as a source of information for different NLP applications, such as answering questions, automatic translation or information retrieval.[2]

Figure 3.1 shows a sentence and it's possible entities.

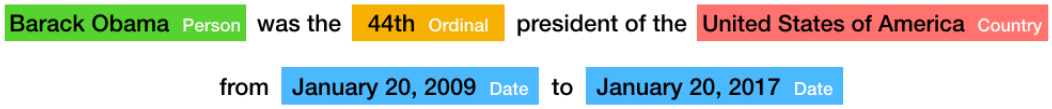

Figure 3.1: Example of entities in a phrase Source: https://github.com/floydhub/named-entity-recognition-template

3.3 Named Entity Recognition on informal Texts

The recognition of named entities in informal texts (eg.: social media posts) is a challenge. This is because they are written inherently differently than formal texts. Unlike formal texts, most social media posts do not follow strict rules, have different grammatical structures, contain misspelled words, informal abbreviations and multilingual vocabulary. They are also relatevely short. For example tweets have a 140 character limit, so not much context can be extracted. For these reasons named entity recognition is a much harder tas on informal texts.

[2]Cf. Behrang Mohit. *Natural Language Processing of Semitic Languages.* 2014, p. 221.

Techniques and tools

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

4.1 Tools

NLTK

The Natural Language Toolkit (NLTK) is a collection of libraries and programs for linguistic applications in the programming language Python.

4.2 Techniques

NER - Process

Figure 4.2 highlights the process of Classifying words given as raw text. Each step will be expanded on in the following subsections.

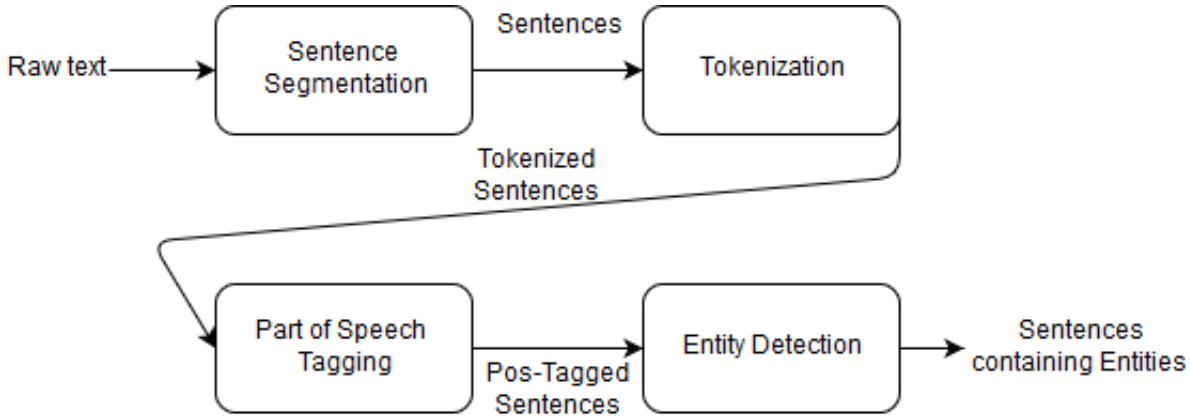

Figure 4.2: Process

Sentence Segmentation

Sentence segmentation is the division of a text into its component sentences.

Word - Tokenization

Word Tokenization describes the process of dividing Sentences into it's component words.

Part of Speech Tagging (POS Tagging)

In POS Tagging each word of a text is assigned a parch of speech token.POS stands for part of speech, and can tell us syntactic information about a word. The tagset used by the NLTK Tagger is the Penn Treebank tag-set.

Penn Treebank Tag Set

The Penn Treebank Tag set uses the tags shown by table 4.1.

Chunking (Entity Detection)

Chunking is a step following POS tagging and structures a sentence into "chunks". Sentences are chunked into the IOB format. Each token is tagged with one of three chunk tags, I (inside), O (outside), B (begin). The B-token marks the beginning of a chunk. All subsequent tokens belonging to the chunk are marked with I. Behind the B and I Tag the chunk type is added. The chunk type describes the type of entity (eg.: Person).

1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Table 4.1: Penn Treebank Tagset

N-gram Chunking

N-gram taggers use a simple statistical tagging algorithm. Each token is assigned the tag that is most likely it's type. It uses so called n-grams which are subsequences of n items. The n amount of previous words and POS-tags are used to guess the tag for the current word. This information is saved inside a context Dictionary maintained by the tagger. A Unigram Tagger doesn't use any context and only uses th POS-tag of the current word to assign a tag. Such a Unigram tagger would for example not be able to differentiate between "the wind2 or "to wind", which are different types of word and have completely different meaning.

Feature Chunking

Often it is not enough to just look at the POS-tag of a words, more information need to be included. This information is determined by a feature extraction-function. One example is shown in listing 4.1.

Listing 4.1: example of feature function

```
def prev_next_pos_iob(tokens, index, history):
    word, pos = tokens[index]

    if index == 0:
        prevword, prevpos, previob = ('<START>',) * 3
    else:
        prevword, prevpos = tokens[index - 1]
        previob = history[index - 1]
```

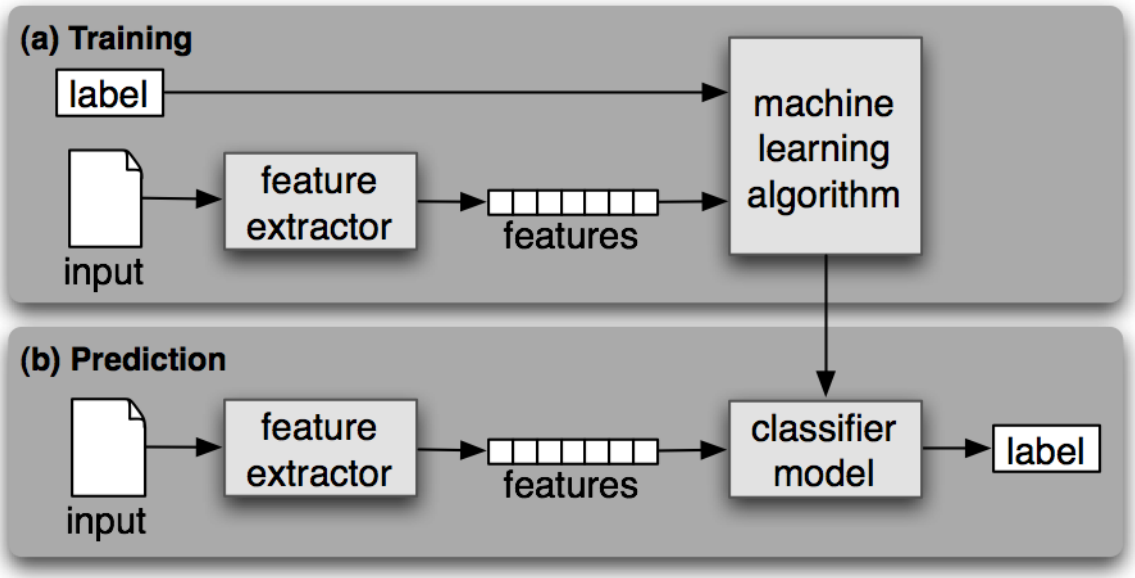

Figure 4.3: Classification Process

source: https://www.nltk.org/book/ch06.html

```
if index == len(tokens) - 1:
    nextword, nextpos = ('<END>',) * 2
else:
    nextword, nextpos = tokens[index + 1]

feats = {
    'word': word,
    'pos': pos,
    'nextword': nextword,
    'nextpos': nextpos,
    'prevword': prevword,
    'prevpos': prevpos,
    'previob': previob
}
return feats
```

Figure 4.3 shows the classification process using such a feature extraction-function. During training each input is transformed into a feature set, which contains the information on which the input is to be classified on. The

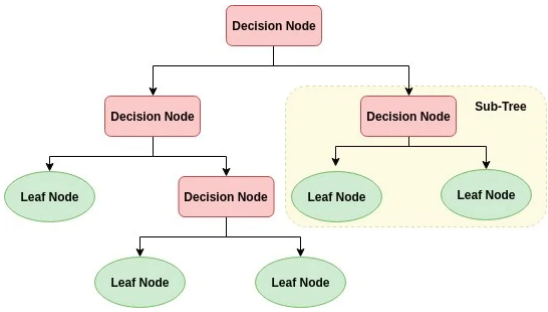

Figure 4.4: Representation of a Decision Tree

source:https://www.datacamp.com/community/tutorials/decision-tree-classification-python

machine learning algorithm takes these feature-sets with their corresponding labels and creates a model.
During prediction the same feature extractor-function is used to convert the input into features, which are then used by the model to predict the corresponding label

Decision Tree A decision tree (4.4) is a flowchart-like structure that can be used for classification. The Tree is made up of decision- and leaf-nodes. Decision nodes correspond to a feature which is checked inside. The process starts at the root leaf and continues through decision nodes until a leaf node is reached, which assigns a label.

The decision tree is build in the training phase through entropy and information gain. Information gain measures how much the organisation of input values increase. To do that the entropy (disorganisation) of their labels has to be calculated. Low entropy would mean that most input values correspond to the same value while high entropy would mean that the labels of the input labels vary widely.

$$entropy = -\sum p(label)logp(label) \tag{4.1}$$

The entropy formula (4.1) shows that either a high amount of inputs from the same label (low $logP(l)$) or a low amount of input from the same label (small $P(l)$) lead to low entropy. While a wide amount of lables with medium frequency would lead to a high entropy.

$$??InformationGain = Originalentropy - newentropy \tag{4.2}$$

The decision Tree is made up by single decision trees called decision stumps. To construct the tree these decision stumps are added. To decide on which feature is used in the first decision node the entropy of all input attributes is calculated. To do that the average entropy of it's leafs is calculated. Then the information gain can be calculated using formula (**??**). The decision tree is then build by selection selecting the decision stumps with the highest information gain.

Discriminative vs Generative Models Discriminative models calculate conditional probabilities ($P(label|input)$) instead of the joint probability ($P(input, label)$ generative models calculate.

NaiveBayes The Naive Bayes Classifier is a discriminative mode and uses an algorithm based on the Bayes Theorem. The algorithm assumes that features are independent of each other (class-conditional independence), meaning that each feature contributes independently towards the probability of a certain result, ignoring possible correlations. (naive)

The Bayes Theorem states the following:

$$posterior = \frac{priori * likelihood}{evidence} \tag{4.3}$$

Knowing that the following equation can be made.

$$P(Class|features) = \frac{P(Class)P(features|Class)}{P(features)} \tag{4.4}$$

Features is a dependent feature vector of size n:

$$features = (f1, f2, f3, ..., fn) \tag{4.5}$$

The Priori is the prior probability of a class. It gets calculated by simply determining the frequency of it in the training set.

Due to the assumed class conditional independence among the features $P(features|Class)$ can be rewritten as such: $P(f1|Class) * ... * P(fn|Class)$

This gets us the following:

$$P(Class|f1, ..., fn) = \frac{P(f1|Class)...P(fn|Class)P(Class)}{P(f1)...P(fn)} \tag{4.6}$$

For all inputs, the denominator remains static. Therefore, it is removed and a proportionality is introduced.

$$P(Class|f1, ..., fn) \propto P(Class) \prod_{i=1}^{n} P(x_i|Class) \tag{4.7}$$

To determine a class the maximum probability needs to be determined:

$$y = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^{n} P(x_i|Class) \tag{4.8}$$

Multiple different naive Bayes Classifiers, making different assumptions regarding the distribution of $P(x_i|Class)$ have been used.

Gaussian Naive Bayes Gaussian Naive Bayes assumes that the likely hood of features is Gaussian.

$$P(x_i|y) = \frac{1}{\sigma_y\sqrt{2\pi}} e^{-(x_i-\mu_y)^2/2\sigma_i^2} \tag{4.9}$$

σ_y and μ_y are estimated using the maximum likelyhood method, which is an estimation methos in which the values are estimated by taking a sample.

Multinomial Naive Bayes Multinominal Naive Bayes asuumes a multinominal distrubution of the data. The vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ parameterize the distrbution for each class y where n is the number of features and θ_{yi} is the probability $P(x_i \mid y)$ of a feature i to appear in class y.

θ_y is estimated using relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{4.10}$$

$N_{yi} = \sum_{x \in T} x_i$ is the number of times a feature i appears in the class y, while $N_y = \sum_{i=1}^{n} N_{yi}$ is the count of all features of the class y. The smoothing parameter α makes sure there are no zero probabilities (eg. Leplace Smoothing).

Maxent Maximum entropy classifiers work in a similar fashion as naive Bayes classifiers. The key difference is that they don't assume that the features are independent. They are discriminative models instead of generative models.

While building the model it's parameters are not determined by probabilities like in the Naive Bayes Classifier, but instead it uses search techniques to find the parameter that maximize the total likelyhood defined by:

$$P(features) = \sum_{x} |in|corpus P(label(x)|features(x)) \tag{4.11}$$

$P(label|features)$ is defined by:

$$P(label|features) = P(label, features)/\sum_{l} abel P(label, features) \tag{4.12}$$

Dependent features can be complex. That is why the model parameters are determined using iterative opimization. The parameters are initialized randomly and then optimized in each iteration. Basically the algorithm for each joint feature the empirical frequency(frequency in training set) of that feature is calculated. >It then searches for the distribution which maximizes entropy, while still predicting the correct frequency for each joint-feature. Not being able to directly calculate the parameters makes training these models time costly

Support Vector Machines Support Vector machine classifiers crate a hyperplane devideing two classes of data. The hyperplane created has the highest distance possible to the nearest training example of both classes, which enables a good seperation. 4.5 visualises such a devision.

Two Differnt concepts are applied to make Support Vectors able to deal with multiple classes: One vs. the rest (as used in scikit LinearSVM) and One vs. one (used in scikit SVC and NuSVC).

One vs the rest Fear each class a binary SVM classifier is trained. One class represents the class wile the other represents all other classes. That means that each classifier determines wheater an example bleongs to it's class or any other class. These classifiers don't only return a class label, but also a confidence score to avoid ambiguity. The class of the classifier that classified it's own class with the highest confidence score is selected.

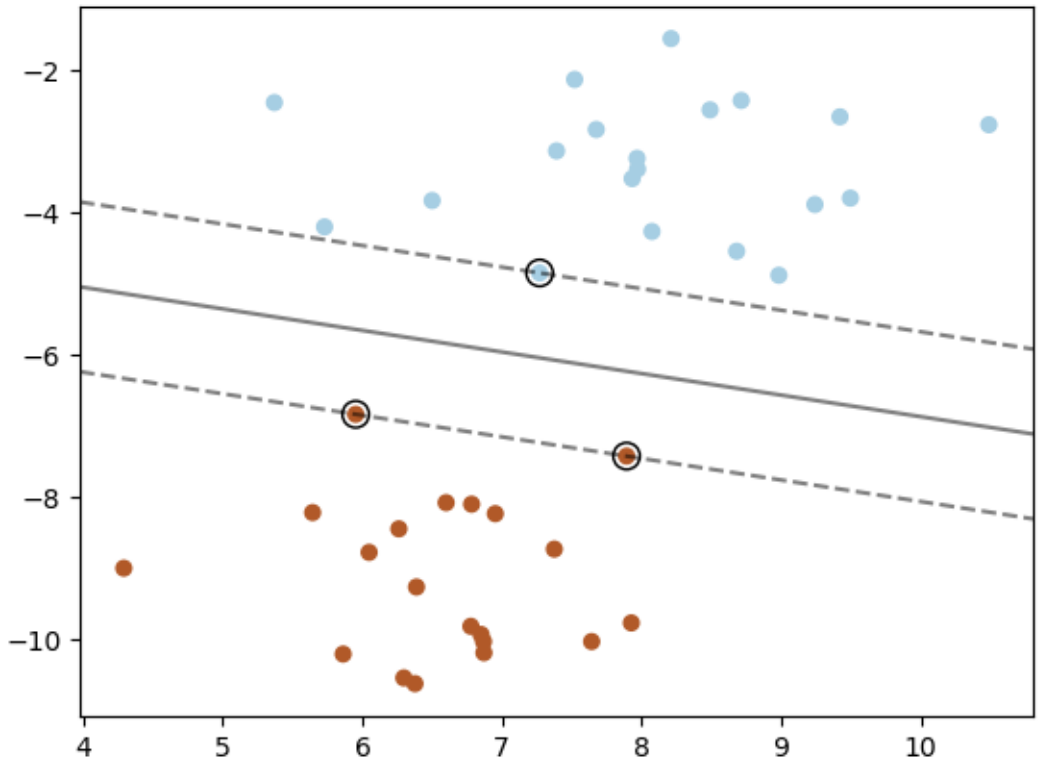

Figure 4.5: Seperation by a hyperplan - Source: https://scikit-learn.org/stable/modules/svm.html

One vs. one A binary classifier is trained for each pair of classes. Each receives the samples of the pairs target classes from the training set, and learns to distinguish these two classes. While predicting a voting procedure is used to combine the outputs.

Gradient Boosting Classifier Gradient Boosting Classifiers are ensemble methods which combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability. The idea of boosting methods is to build several weak clasifiers and to combine them into a strong one.To do that base estimators are beeing build sequentially and then added to a combined estimator to reduce bias. Gradient Boosting Classifiers sequentially build base classifiers to predict the labels of samples,

and calculate the gradient of the loss function in regards to the predection (difference between the outcome of the learner and the real value).

Random Forest Classifier Another ensemble classifier is the Random Forest Classifier. The esnamble is made up by randomized decision trees.

Each tree is constructed with a bootstrap sample (replacement sample). The decision trees are build shlightly differntly to the classical decison trees. Instead of considering all features at the splits only a random subset of features is considered. This randomness results in a higher bias of individual trees.

But since the prediction of the ensemble is given as the averaged prediction of the individual classifiers this increased bias is usualy more than compensated, yielding a better result.

Extremely Randomized Trees Classifier Extremly Randomized Trees are a variance of Random Forest classification in which the randomness is even higher. When choosing a random subset of features

"As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias:" ???????

Relevant aspects of the development of the project

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Related works

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusions and future lines of work

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliography

[1] Jason Collobert Ronan; Westan. *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*. NEC Labs America. URL: https://ronan.collobert.com/pub/matos/2008_nlp_icml.pdf.

[2] Behrang Mohit. *Natural Language Processing of Semitic Languages*. 2014, p. 221.