

ALMA MATER STUDIORUM · UNIVERSITÀ DI  
BOLOGNA

---

SCUOLA DI SCIENZE  
Corso di Laurea in Informatica

**Entity extraction e text classification  
come strumenti per la sentiment  
analysis di testi giornalistici**

**Relatore:**  
Chiar.mo Prof. Fabio Vitali

**Presentata da:**  
Martino Pizzol

**Correlatori:**  
Chiar.mo Prof. Edoardo Mollona  
Dott. Angelo Di Iorio  
Dott. Luca Pareschi

**Sessione 3**  
**Anno Accademico 2013/2014**

# Introduzione in Italiano

Questa tesi si basa sull'utilizzo di nuove tecnologie per facilitare l'analisi di articoli giornalistici e pone l'attenzione, in particolare, su algoritmi di *sentiment analysis* e di *semantic text classification*.

Lo scopo finale di questo lavoro consisteva nell'aiutare i ricercatori sociali automatizzando i processi di analisi di grandi quantità di documenti.

Nelle Scienze Sociali alcuni ricercatori usano la *teoria dei frames* per svelare i significati nascosti nella realtà che viene descritta spesso in maniera complessa e dettagliata dai quotidiani. Questo metodo consiste nell'analizzare un set di documenti redigendo poi una mappa di classificazione che sarà utilizzata in seguito per classificare i testi, e costituirà la base di partenza dei ricercatori per la dimostrazione delle loro ipotesi.

Attualmente molti algoritmi sono in grado di classificare testi automaticamente; nello specifico essi possono individuare la categoria di appartenenza di un testo basandosi sulle informazioni contenute nei modelli di classificazione. È importante specificare che un "modello di classificazione" descrive delle categorie mediante delle parole chiave. Per esempio, se volessimo classificare una serie di articoli stabilendo se riguardano lo sport o la politica, potremmo creare un modello di classificazione che descrive rispettivamente le due categorie con le parole chiave ad esse correlate (ad esempio per la politica si potrebbe fare riferimento a parole quali: *Partito Democratico* e *Forza Italia*, mentre per lo sport si potrebbero considerare le parole: *pallacanestro* e *pallavolo*). Infatti l'idea fondamentale di questi algoritmi è quella di indicare quali categorie appaiono in un testo facendo riferimento alle parole chiave in

esso presenti.

Nel nostro caso una famiglia di algoritmi molto importante è quella relativa al *topic modeling*, il quale, analizzando un insieme di documenti testuali riesce a classificarli automaticamente creando vari modelli di classificazione che possono essere utilizzati dai ricercatori per trarre nuovi spunti di riflessioni per le loro ricerche.

Questi strumenti sono estremamente potenti perchè permettono ai ricercatori di trovare degli aspetti su cui non avevano mai ragionato, non sempre però riflettono esattamente le loro idee. Non avere a disposizione uno strumento che permetta loro di classificare dei documenti secondo le idee che hanno già formulato, in certi casi, costituisce un limite.

A questo punto, confrontando il modo di classificare dei computer con quello umano, possiamo capire che l'uomo abbina ad ogni parola un significato in base al contesto ed alle conoscenze pregresse. In merito possiamo citare Ausubel, il quale afferma che: "Il singolo fattore più importante che influenza l'apprendimento sono le conoscenze che lo studente già possiede".

Sono stati creati quindi degli algoritmi di classificazione cercando di utilizzare la conoscenza umana per comprendere automaticamente il giusto significato di ogni parola. In particolare è nato un filone di ricerca che utilizza Wikipedia come fonte di conoscenza. Più precisamente ogni pagina in essa contenuta viene considerata dal sistema come un concetto base (per esempio può rappresentare una persona, un luogo, un evento o un oggetto) ed i collegamenti ipertestuali vengono interpretati per comprendere come i concetti sono legati tra loro. Questi algoritmi, quindi, analizzando un testo sono in grado di abbinare alle parole il giusto significato collegandole, dove possibile, alla relativa pagina di Wikipedia. Tale sistema trae giovamento dal fatto che Wikipedia viene sempre aggiornata da persone che vi riversano le proprie conoscenze.

Possiamo così affermare che alcuni algoritmi riescono a classificare il testo utilizzando parte della conoscenza umana, tuttavia, in alcuni casi non riescono ad evincere esattamente il significato di ogni parola. Se per esem-

pio pensiamo a due articoli: uno tratto dal *Manifesto* e l'altro dal *Sole 24 Ore*, dobbiamo considerare che tali testi potrebbero parlare dello stesso argomento utilizzando toni totalmente diversi. In questo caso sarebbe utile far riferimento agli algoritmi di *sentiment analysis* che sono in grado di capire se una frase esprime un sentimento positivo, negativo o neutro. Utilizzando questi algoritmi possiamo infatti contestualizzare meglio ogni frase in modo da creare un modello di classificazione adatto e migliorare così il risultato finale.

In questa tesi è stata sviluppata un'applicazione web chiamata SAClet, che permette ai ricercatori sociali di importare le analisi effettuate con la suddetta *teoria dei frame* per poi riutilizzarle nella classificazione di nuovi documenti.

Il sistema è in grado quindi di interpretare il lavoro dei ricercatori generando un modello di classificazione che potrà essere valutato e migliorato al fine di classificare in maniera corretta nuovi documenti.

Per testare le funzionalità di SAClet, ho utilizzato una ricerca effettuata dal professore Edoardo Mollona e dott. Luca Pareschi, riguardante la privatizzazione delle acciaierie italiane negli anni '80. Nello svolgimento di tale ricerca sono stati analizzati seicento articoli creando un insieme di *frames* che il sistema ha poi utilizzato per generare un modello di classificazione in grado di analizzare altri documenti.

I risultati di classificazione così ottenuti sono soddisfacenti, specialmente se si considera il fatto che i modelli di classificazione utilizzati sono stati generati automaticamente dal sistema; chiaramente l'intervento umano potrebbe migliorare notevolmente le prestazioni.

Un altro aspetto molto interessante, è stato il fatto che il sistema ha rilevato dei problemi di sovrapposizione delle categorie del modello prima ancora che se ne accorgessero i ricercatori. Questo fatto ci fa pensare che lo strumento è un ottimo banco di prova anche per valutare il loro lavoro.

Le prestazioni del sistema sono state penalizzate anche dal fatto che gli articoli presi in considerazione erano scritti in italiano e quindi gli algorit-

mi di classificazione erano basati sulla Wikipedia italiana, che risulta essere molto meno dettagliata rispetto a quella inglese. A tal proposito sarebbe interessante provare il sistema su un insieme di documenti scritti in inglese.

Purtroppo nel sistema non è stato incluso nessun algoritmo di sentiment analysis, che sicuramente lo avrebbe aiutato ad utilizzare dei modelli di classificazione adatti al contesto di ogni frase.

Il lavoro effettuato fin'ora mette le basi per poter iniziare ad utilizzare i piú moderni algoritmi di *sentiment analysis* per migliorare ulteriormente l'analisi del contesto dei documenti.

Stiamo già lavorando per implementare nuove funzionalità. Infatti la pagina che visualizza i risultati della classificazione di un corpus di articoli attualmente permette una navigazione basata sulle categorie rilevate; siamo già al lavoro per crearne una nuova incentrata sui concetti estratti dai testi o sulle co-occorrenze con le categorie stesse. Allo stesso modo stiamo progettando dei filtri di ricerca che sfruttano i collegamenti tra i concetti permettendo, ad esempio, di trovare tutti i documenti in cui sono citati il Presidente della Repubblica ed il suo predecessore.

Vogliamo integrare in SAClet qualche algoritmo di *topic modeling*, in modo tale da permettere ai ricercatori di caricare dei set di articoli che non hanno mai visto, per ottenere automaticamente nuovi modelli di classificazione suscettibili di ulteriori miglioramenti.

Questo sistema renderebbe SAClet una sorta di lente di ingrandimento per poter sondare i significati nascosti nei testi, leggendo attraverso le parole.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Content analysis</b>	<b>6</b>
2.1	Qualitative research in social sciences . . . . .	6
2.2	Can computer science help qualitative researchers? . . . . .	7
2.3	Topic Modeling and Article Classification . . . . .	8
<b>3</b>	<b>SAClet: a tool for document classification</b>	<b>10</b>
3.1	Main workflow . . . . .	10
3.2	The web application . . . . .	13
3.2.1	Handle classification model . . . . .	13
3.2.2	Documents classification . . . . .	16
<b>4</b>	<b>SAClet internals</b>	<b>20</b>
4.1	Application structure and technologies . . . . .	20
4.1.1	Deploy . . . . .	25
4.1.2	Scalability . . . . .	26
4.2	Algorithms . . . . .	27
4.2.1	Classification models generation . . . . .	27
<b>5</b>	<b>Evaluation</b>	<b>30</b>
5.1	Classification models . . . . .	30
5.2	Documents classification . . . . .	36
<b>6</b>	<b>Conclusions</b>	<b>38</b>

# Chapter 1

## Introduction

In this thesis we are going to talk about technologies which allow us to approach sentiment analysis on newspapers articles. The final goal of this work is to help social scholars to do content analysis on big corpora of texts in a faster way thanks to the support of automatic text classification.

In social sciences, some qualitative researchers use the *frame analysis methodology* to discover obscure relationships hidden in large sets of documents. This methodology consists in analyzing a set of articles, creating a coding sheet and using it to classify all documents; thanks to the classification thus obtained, scholars can support their initial hypothesis.

There are many algorithms which can classify texts in an automated way. In facts they are able to decide the category a text belongs to by using the knowledge contained in classification models.

A classification model is basically composed of key concepts, called categories, which are represented by a set of relevant words. For instance, if we want to classify some news, and we want to tell whether they concern sports or politics we can create a classification model that describes the two categories with their relevant words (like football and basketball for sports, democrat and republican for politics). The basic idea of these kinds of algorithms is to tell which categories appear in a text understanding which set of relevant words is present.

Another interesting family of algorithms is topic modeling, which can classify sets of documents using auto generated classification models. This gives scholars many new hints to understand and interpret documents. In fact these tools are very efficient and innovative, but that does not necessarily mean they reflect the work researchers had previously done.

If we compare the performance of these tools to human approach in text classification, it is clear that humans are conditioned by their cultural background. Actually as Ausubel said “*The most important single factor influencing learning is what the learner already knows.*” [ANH<sup>+</sup>68].

In order to obtain a similar effect, several algorithms have been developed which use human knowledge to interpret the meaning of every word present in a text. In fact, a specific research branch started using Wikipedia as a knowledge base, in which researchers developed algorithms which were able to understand the meaning of any single word linking it to its respective Wikipedia page. The identified word takes the name of the *named entity*, which could refer to persons, places, events or material things. The links between all Wikipedia pages help the algorithm to better understand the context. Finally we can observe that this knowledge base is always evolving because many people contribute to keep it up to date.

With this in mind some algorithms can classify texts implicitly using parts of human knowledge. Again we can find some cases where software fails to understand the right meaning of a word; let’s think about politically oriented newspapers which can talk of the same topic yet give words different meanings which can be positive, negative or neutral. To let software deal with these nuances we can use sentiment analysis algorithms and in this way we can help classification algorithms by giving them some hints to better understand the context.

The main idea behind my work is to use semantic technologies supported by other technologies like the sentiment analysis in order to better understand and classify newspaper articles.

*SAClet* is a web application that has been developed to help researchers



import the work they have previously done so that they can classify any new sets of documents. The classification models automatically generated reflect the results of scholars' previous research.

Not only does the software provide researchers with tools able to evaluate the efficiency of their classification model but it also gives them the possibility to enhance it with their specific knowledge of the topic.

Furthermore researchers are given the chance to analyze any kind of texts, even out of a set of documents, classifying each paragraph and explaining which topics were involved in the choice of classification.

This system has been tested by some qualitative scholars who have analyzed about 600 articles concerning the privatization of Italian public steel industries. In this case the system created a classification model in an automated way by relying on researchers' work in order to enable them to try this classification on a new set of articles.

The results obtained with the classification model generated by the system without researchers' improvements was good, yet the most interesting outcome was the evaluation of this model. Actually SAClet found some problems concerning overlapping categories which in the meanwhile had also been discovered also by researchers.

Nevertheless the classification performance was disadvantaged by the fact that texts were written in Italian and Wikipedia's Italian version is not as rich in links as the English one. The usage of English texts would improve the system's overall performance, but out of the box it works also with French, German and Portuguese texts.

The work we have done so far is a springboard to enhance text classification using advanced sentiment analysis algorithm to improve the context detection.

At the moment we are creating an innovative interface to explore any sets of documents by the following criteria: text classification, extracted named entities and their co-occurrences. Furthermore named entities permit us to create smart filters based on Wikipedia knowledge; for instance, if we

consider the visualization of all documents regarding a Prime minister and his predecessor.

Furthermore we are developing a way to combine semantic classification algorithms with the topic modeling ones. SAClet could then propose to scholars the power of both instruments, letting them upload any set of documents and obtain some classification models which are ready to use. This way SAClet could become a sort of magnifying glass for big corpora of texts.

The structure of the thesis is the following:

**Chapter 2** is an overview of the qualitative researchers' work and on text classification technologies;

**Chapter 3** is an overview of SAClet, the web application developed in this thesis;

**Chapter 4** gives a technical description of SAClet's architecture and functioning;

**Chapter 5** contains an evaluation of the results obtained during the development of SAClet;

**Chapter 6** concludes this thesis and proposes some hints about future works.

## Chapter 2

# Content analysis: qualitative and automated extraction of meaning

### 2.1 Qualitative research in social sciences

Traditionally there are three different ways to approach qualitative research in social sciences [DNB13]. The first one consists in reading texts and interpreting them directly. Non replicability is the main drawback of this method; in facts for other scholars it is impossible to apply the same interpretation on other articles.

The second one uses a lively and important methodology called *frame analysis* [Mat09]. Yet this method too is not “trouble free”, we can easily detect three main problems in it. First of all it is very difficult to analyze large sets of articles. Secondly if a scholar discovers too many frames the final analysis can be hard to do. Thirdly, scholars must have an idea of what they want to find before starting.

In the third one scholars use computers to discover whether a specific set of words appears in a text or not. This method is mainly based on a researcher intuition and it implies, as a drawback, the fact that some hidden

meaning could be totally ignored.

The qualitative research that I used as a case study is the one made by professor Edoardo Mollona and dr. Luca Pareschi who followed the methodology of frame analysis.

In their research they started studying the context of steel privatization in Italy identifying a set of relevant keywords.

They used these keywords to extract a set of articles from the online archives of two of the most important Italian newspapers: *La Repubblica* and *Il Sole 24 Ore*; unfortunately they could not take articles from another famous newspaper called *Il Corriere della Sera* because its archives were not completely available online. Researchers chose articles coming from all sections of newspapers covering the period from 1984 to 1995; the set was composed by 537 articles, 246 coming from *La Repubblica* and 291 coming from *Il Sole 24 Ore*.

At this point they started reading all articles to extract the most relevant ones; they obtained 1902 statements that were then processed in order to create nodes using a methodology which is well explained by Matthes [Mat09]. The resulting nodes are collections of statements concerning the same concept.

Researchers used then these nodes and connected them into storyboards called frames which are an interpretation schema of reality [Gof74]; thanks to this process they obtained 12 frames which provided them a tool to classify every original article and proof their hypothesis.

Meyer and Höllerer [MH10] have created a map of shareholder value in Austria using the same methodology.

## 2.2 Can computer science help qualitative researchers?

As you can imagine the some steps of frame analysis methodology are really demanding since researchers have to read and analyze entire sets of

documents; of course, their effort increases when they have to deal with large sets of articles. The advantages of an automatized research would be considerable because it would accelerate researchers' work.

This topic has already been considered in a work by Di Maggio [DNB13] in which he identifies four characteristics an ideal automated system must have.

First of all it must be explicit in the sense that it must allow researcher to test their interpretations and at the same time it must be available to other researchers.

Secondly it must be automatic and handle a big amount of documents with no need of human input.

Thirdly it must be inductive, which means it should let researchers discover the set's structure before the bases of analysis are arranged; furthermore it must allow other researchers to use the same set for further researches.

Fourthly it must be able to understand the text's context in order to deduce the right meaning of words and avoid issues related to polysemy.

## 2.3 Topic Modeling and Article Classification

Topic modeling algorithms are a bridge linking social sciences to computer science; first they are able to analyze a set of documents in order to extract relevant topics composed by a list of keywords. Finally these algorithms classify the input documents referring to the topics they found [Ble12]. In this way scholars only have to interpret the meaning of each topic and can analyze large sets of documents.

These algorithms are really precious instruments for researcher because they propose them various sets of topics to work with.

The University of Massachusetts-Amherst developed a tool called MALLET [McC02] including the most powerful topic modeling algorithms like: textitLatent Dirichlet Allocation (LDA) [BNJ03], *Pachino Allocation* [LM06] and *Hierarchical LDA* [BGJT04]. In fact MALLET is the software thanks

to which researchers can benefit of topic modeling algorithms, it is a sort of magnifying glass that shows hidden meanings of big sets of documents.

Yet MALLET is not able to use any sets of frames previously discovered by researchers. This can be a limit for scholars who have just applied the *frame analysis* on a set of documents and want to convey the results on other ones.

Furthermore on short texts with small context the software is likely to be mistaken because it doesn't use words' semantics; this behavior has been found in [DNB13] and it can introduce noise analyzing short newspaper's articles.

The system I imagine thus, should be able to take advantage of the results of frame analysis done by scholars and manage to classify texts using named entities instead of simple words.

To do this now a days there is a line of research using Wikipedia as the main knowledge base. From this perspective Wikipedia is an incredible source of knowledge for the machines [WD08] that allows to create classification models very close to the human knowledge.

In these systems every Wikipedia page represents a named entity and the links between pages are useful to understand the entities connections; in this way, computers can compute the relatedness of the entities [WM08], an essential concept to compare entities during the classification phase.

Dandelion DataTXT<sup>1</sup> is a set of text analytics API that uses Wikipedia as knowledge base. DataTXT-NEX lets users extract named entities [FS10] from any texts, even the shortest ones. DataTXT-CL allows users to build classification models using named entities and classify [VFS12] any kind of texts using these models.

---

<sup>1</sup><https://dandelion.eu/datatxt>

# Chapter 3

## SAClet: a tool for document classification

This chapter contains an overview on SAClet which goes from a general description of the entire workflow to the details of users interaction.

### 3.1 Main workflow

Traditionally, content analysis in social science research has been a very time consuming activity. Researchers use to code their data manually, or through the use of software, such as Nvivo, that helps them codifying texts, but does not automatically extract meanings. This qualitative kind of research provides scholar with meaningful results, but it is hard to replicate and to conduct on big corpora of texts.

I started using a research done by professor Edoardo Mollona and dr. Luca Pareschi in which they analyze the public debate to understand how newspapers have reported the Italian steel industry privatization.

At the very beginning they have collected and analyzed a set of articles in order to create a set of frames which are an interpretation schema of reality [Gof74]. For instance in the *Neo Liberalism* frame (NL) privatizations is seen as a positive cultural revolution that will improve the Italian economic system

while in the *Labour versus Capital* frame (LSVC) privatization appears as potentially dangerous for the occupation.

This set of information is a precious starting point to create classification models automatically; to the system this is the users' knowledge base. This process is represented in figure 3.1

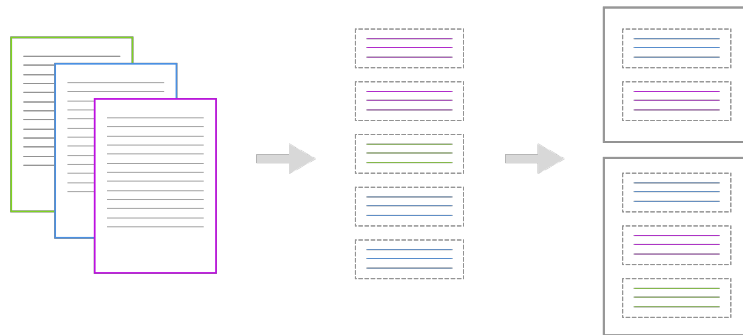


Figure 3.1: How qualitative researchers analyze sets of documents

During the *import phase* the system enhances the knowledge base extracting named entities. A named entity could be a person, a place, an event or a material thing and the most important fact is that it is linked to a wikipedia page.

At this point the system has got all the informations it needs to create automatically a classification model starting from the users' knowledge base. Thus from each frame the system selects a set of references from which it extracts a set of weighted named entities; then it filters them using a *td-idf* method in order to keep only the most representative ones. The output of the *creation phase* is a classification model composed by a set of frames containing named entities whose weight represents their importance.

In order to use a more standard labelling the system calls this frames categories and the named entities topics. In my opinion this is useful because users can easily find the same labelling in other classifiers.

The classification model has a dedicated web page where the user can view, edit and test it. Some categories can be poor in topics but users can



add new ones with a smart autocomplete which is powerful because it is based on wikipedia page relations. This feature enables users to find the desired topic without any kind of knowledge on wikipedia labelling.

Users can evaluate the classification model on the references the system has not used during the creation phase. In that case the output is composed by the classical parameters to judge a classifier: recall, precision and f-score. Recall is the ratio between the number of right matches retrieved by a research and that of all correct existing documents. Precision is the ratio between the number of right matches retrieved by a research and that of all documents retrieved from the same research. F-score is a global measure defined as the harmonic mean between precision and recall. To better understand the classifier performance, the system provides also a confusion matrix. Each row of the matrix represents instances of predicted categories while the column represents instances of found categories, this allows users to get a more detailed analysis.

After this training phase the classification model is ready to be used on a set of documents. Users can upload a zip file with all the documents and classify it.

The system analyzes each document splitting it in paragraphs which are classified using a threshold in order to exclude poor results; finally it combines all outcomes creating a final document classification. In this way not only can the system show all the categories present in each document but it can also visualize their coverage.

The entire classification is available in a page that permits to filter documents that contain a specific category. Nevertheless the system gives the users more information about the classification results of each page, so users can see how the system classifies each paragraph.

Besides in this page users can classify the text using every classification model available in the system and submit any other document just to try a shot classification.

## 3.2 The web application

Users can thus create classification models and classify sets of documents using only a web application, without knowing what really happens under the hood. First of all I am going to explain the user experience.

### 3.2.1 Handle classification model

The first step for the user is to create a new classification model; to do this the system provides a form asking some basic information (like name and description) and some more advanced ones (like the size of each category, and a flag to use the most relevant name entities).

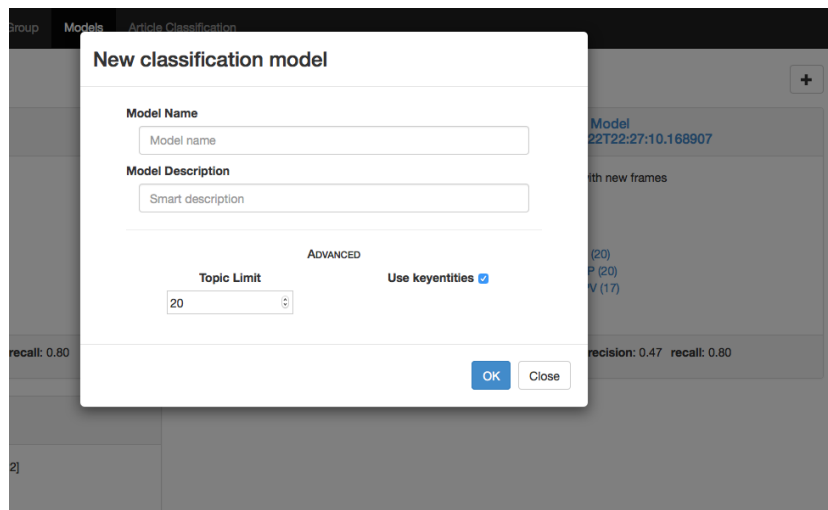


Figure 3.2: Modal form to create new classification models

The system then generates a classification model and shows it in a specific web page. This page is very complex because we must observe two strict and opposite constraints: on the one hand we must show a lot of information; on the other hand it is very important to have an overview of it. The first version of this page displayed a lot of information for each topic but lacked an overview of all categories; for that reason it was difficult to understand the right category each topic belonged to.

The final solution exploits each page dimension so that: height displays all topics and width displays all categories. I think this is the best compromise for the page usability because we can give the users an overview of the model and an easier way to do some bulk operation like delete. For auto generated models this feature is very precious because users can see the system's mistakes and fix them in an efficient way. An example of this page is showed in figure 3.3 where you can see:

- the classification model called *Protezionismo KE*,
- the categories created by the system from a user's knowledge base such as *PR, NL, CC...* ,
- lists of topics for each categories.

From this page users can see each topic's details (description, Wikipedia and DBpedia links, pictures and wikipedia categories), modify its weight or remove it.

**Protezionismo KE** version with keyentities edit  test results

PR	NL	CC	LVSC CVP	SI SL CV
Nazionalismo	Libertà	Economia mista	Federazione Italiana Metallmeccanici	Boiardo (s
Innovazione	Mercato finanziario	Corrigiano	Sciopero	Partitocrz
Telit	Domanda e offerta	Monopolio	Ultim	Storia anti
Aistom	Qualità	Know-how	Servizio militare	Sergio Pini
Autorizzazione (diritto)	Mercato	Ministri dello Sviluppo Economico della Repubblica Italiana	Federazione Impiegati Operei Metallurgici	Fiume
Documento	Bene (etica)	Prodotto (commercio)	Bagnoli (quartiere di Napoli)	Prima Rep
La matassa	Pluvalenza	Divisione del lavoro	Vittorio Lavezzari	Partito poli
Masochismo	Luce	Energia	Occupazione	Democraz
Consiglio europeo	Proprietà (diritto)	Finsider	Dalmine	Innocenzo
Colonia (insediamento)	Risparmio	Mercato	Unione Italiana del Lavoro (1950)	Partito Nis
Siemens (azienda)	Statalismo	Bagnoli (quartiere di Napoli)	Confitarma	SISMA
Azionista	Homo sapiens	Varese	Stato di ossidazione	Provincia c
Establishment	Petrolio			Legittime c

Figure 3.3: Classification model page

In order to improve the classification model users can add other topics to each category enhancing its semantic coverage; the problem is that nobody

knows the title of each wikipedia page. For example if you want to add the common Italian trade union called *UIL*, you cannot know that the related page on wikipedia appears under the name *Unione Italiana del Lavoro*.

To solve this problem we use a dandelion API that tries to provide a smart autocomplete. This is a powerful solution because it is based on wikipedia page relations; to better explain to users each topic the system shows a picture and a small description of it. This feature enables users to find the desired topic without any kind of knowledge on wikipedia labelling.

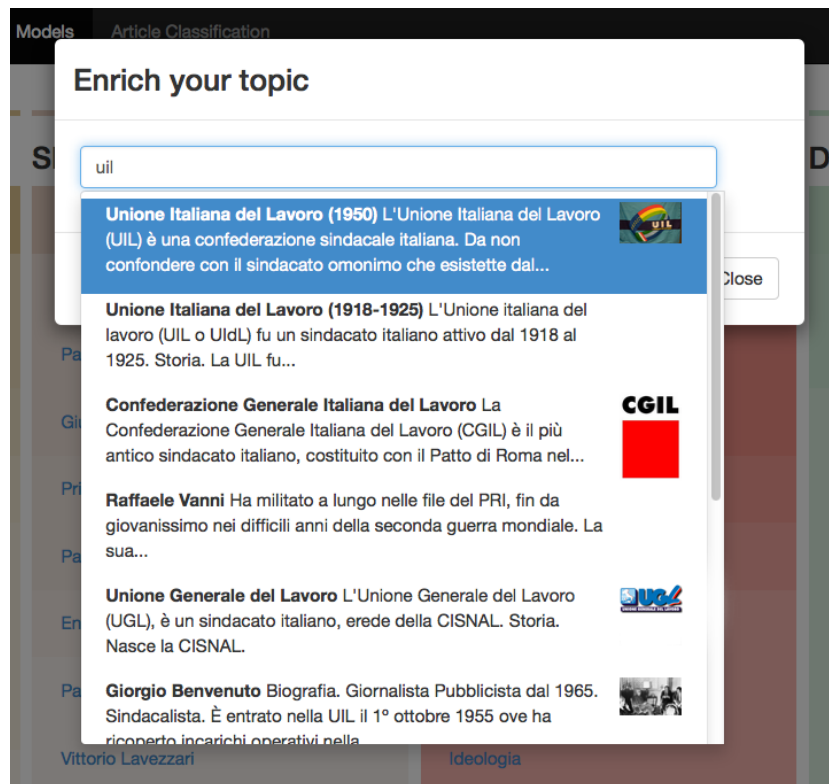


Figure 3.4: Smart autocomplete to expand categories with more topics

For each classification model the system provides a way to judge the performance of the classifier; any time users launch an evaluation the system automatically saves a version for this model and provides a set of revisions to easily revert. The evaluation process runs in a separate task to allow users to keep on using the system and once it is complete all results are available

in a dedicated page.

In this page (which is visible in figure 3.5) the system provides an overview of all evaluations showing the score attributed to each one and allowing the reloading of a specific version. This basic versions manager is transparent to users but let them change a classification model in a safe way because it is always possible to go back to any previous version. The score of each classification model is composed by: precision, recall and f-score; for them the system provides a micro and a macro version. The macro version gives to each category the same weight independently on its frequencies but it is influenced by the classifier's performance on rare categories. The micro version instead gives equal weight to each document and it is dominated by the classifier's performance on common categories.

Another fundamental tool present in this page is the confusion matrix; it is the only way to understand where a classifier fails. Each row of the matrix represents instances of predicted categories while the column represents instances of found categories, in each intersection users can find the percentage of documents and have a view of the most influent topics.

Actually it lets users see which categories are too much generic and hide more specific ones; users have two ways to solve this issue: remove too much generic topics from a category or adjust their weight and add them to other categories.

With some small changes the system allows users to create an arbitrary model which is not based on a knowledge base. In this way without an initial set of classified texts, the system cannot test the created model to create the confusion matrix, this is anyway a very powerful method that lets experiment a classification idea on any set of documents.

### 3.2.2 Documents classification

Once the user has created his classification model, the system permits him to classify any set of text documents and explore the results.

To classify texts users have first of all to import them into the system;

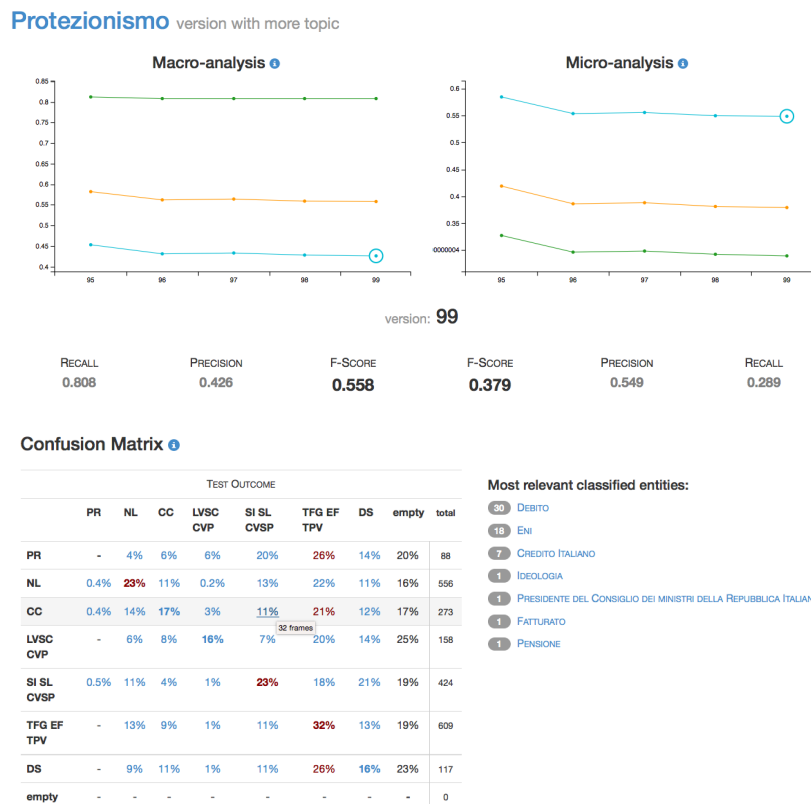


Figure 3.5: Classification model's evaluation page

the real advantage of this kind of tools is that it makes it very easy to classify large set of documents.

To simplify the documents' upload the system can import a zip file from any public URL (a service like dropbox permits to upload a large set of documents and share them). I have avoided the direct file upload from the browsers because with big files it is easy to encounter several problems. In the system this document sets are named *document group* and you can see how they appear to the users in figure 3.6.

Users are able to launch classification on the document sets with a simple form where they can specify which classification model they want to use; they can also set a threshold to filter low scoring classification cases.

To classify an article the system splits it into paragraph, classify each

**Protezionismo Document Set**

<p>01-02-1994 SS940201017AAA.txt 01-02-1995 SS950201013AAA.txt 01-03-1995 SS950301025BAA.txt 01-04-1986 SS860401003CAA.txt 01-04-1994 SS940401001AAA.txt 01-06-1993 SS930601031CAA.txt 01-08-1993 SS930801012BAA.txt 01-09-1992 SS920901023AAA.txt 01-11-1991 SS911101010CAA.txt 01-12-1992 SS921201025AAA.txt 02-03-1993 SS930302029BAA.txt 02-03-1995 SS950302025DAA.txt 02-06-1986 SS860602004AAA.txt 02-06-1988 SS860602004BAA.txt 02-06-1994 SS940602012DAA.txt 02-06-1994 SS940602014AA.txt 02-06-1995 SS950602032AAA.txt 02-07-1993 SS930702029AAA.txt 02-08-1986 SS860802014AAA.txt 02-08-1986 SS860802014CAA.txt 02-08-1986 SS860802014EAA.txt 02-08-1994 SS940802017CAA.txt 02-10-1992 SS921002027AAA.txt 02-11-1988 SS881102019CAA.txt 02-11-1993 SS931102029AAA.txt 02-11-1993 SS931102029CAA.txt 02-12-1987 SS871202012FAA.txt 03-01-1989 SS890103007JAA.txt 03-03-1995 SS950303029BAA.txt 03-03-1995 SS950303029CAA.txt 03-03-1995 SS950303029DAA.txt 03-04-1992 SS920403011AAA.txt 03-10-1995 SS951003015CAA.txt</p>	<p>18-12-1993 SS931218001BAA.txt 19-04-1994 SS940419029DAA.txt 19-05-1994 SS940519025DAA.txt 19-05-1995 SS950519025AAA.txt 19-06-1993 SS930619010AAA.txt 19-08-1993 SS930819015DAA.txt 19-10-1990 SS901019029AAA.txt 19-11-1992 SS921119024AAA.txt 19-11-1993 SS931119001DAA.txt 20-01-1987 SS870120002CAA.txt 20-01-1995 SS950120029CAA.txt 20-02-1993 SS930220025GAA.txt 20-03-1993 SS930320029DAA.txt 20-06-1995 SS950620015BAA.txt 21-01-1986 SS860121011BAA.txt 21-01-1994 SS940121030CAA.txt 21-02-1993 SS930221001HAA.txt 21-02-1993 SS930221013AAA.txt 21-03-1993 SS930321013AAA.txt 21-07-1993 SS930721029CAA.txt 21-11-1991 SS911121002CAA.txt 22-02-1995 SS950222007BAA.txt 22-02-1995 SS950222011AAA.txt 22-03-1991 SS910322011AAA.txt 22-04-1987 SS870422017AAA.txt 22-04-1993 SS930422001GAA.txt 22-04-1993 SS930422025AAA.txt 22-05-1993 SS930522007JAA.txt 22-07-1992 SS920722009BAA.txt 22-09-1993 SS930922009AAA.txt 22-10-1993 SS931022010AAA.txt 22-11-1991 SS911122001DAA.txt 22-11-1993 SS931122029AAA.txt</p>	<p>C 11 dicembre 1987 4.txt C 11 dicembre 1993 47 2.txt C 11 dicembre 1993 47.txt C 11 dicembre 1993 48.txt C 11 febbraio 1994 44.txt C 11 giugno 1994 43.txt C 11 luglio 1993 41.txt C 11 ottobre 1991 1.txt C 11 settembre 1993 43.txt C 12 agosto 1993 43.txt C 12 aprile 1991 37.txt C 12 dicembre 1992 45.txt C 12 febbraio 1992 11.txt C 12 febbraio 1992 45.txt C 12 giugno 1987 7.txt C 12 marzo 1993 43.txt C 12 novembre 1993 23.txt C 12 ottobre 1989 10.txt C 12 ottobre 1991 48.txt C 12 ottobre 1993 1.txt C 13 agosto 1987 41.txt C 13 agosto 1992 1.txt C 13 gennaio 1987 51.txt C 13 gennaio 1993 11.txt C 13 marzo 1987 9.txt C 13 marzo 1991 45.txt C 13 novembre 1992 4.txt C 13 ottobre 1988 45.txt C 13 ottobre 1994 51.txt C 13 settembre 1991 3.txt C 14 aprile 1987 54.txt C 14 aprile 1993 43.txt C 14 aprile 1994 44.txt</p>
---	--	---

599 documents

Add documents

Classify documents

**Classifications**

- 15-02-09 16:39
- 15-02-07 20:50
- 15-01-31 19:34
- 15-01-31 18:21
- 15-01-31 18:20
- 15-01-31 18:11
- 15-01-31 18:10
- 15-01-31 18:07
- 15-01-30 23:52
- 15-01-30 23:41
- 15-01-28 22:54

Figure 3.6: The view of a document group

of them and compute the final scoring filtering the results under a specific threshold.

When the classification ends a link appears on the document group page to see the outcomes. The results' visualization allows users to filter documents on the base of each category and drill down to each classification as you can see in figure 3.7. The biggest advantage of this page is that users can see how the system categorizes each paragraph and its relative named entity extracted. Users can also categorize this page with the other classification models present in the system and immediately see the results.

Protezionismo Document Set / 15/02/09 16:39 / 02-06-1988 SS880602004BAA.txt

Select the classification model

Ultimo Protezionismo

Show model

**02-06-1988 SS880602004BAA.txt**

NL 4% LVSC CVP 14% TFG EF TPV 22%

Le condizioni per buoni accordi Stato-privati SOLO TRA EGUALI JOINT-VENTURE A PROVA DI FLIRT  
Riccardo Gallo  
Nel suo commento su «Il Sole-24 Ore» del 4 maggio scorso, Carlo Mario Guerci ha proposto che le Partecipazioni statali ai vari livelli gerarchici (ministero, ente, azienda) allarghino la collaborazione con l'impresa privata attraverso la costituzione di società miste, le cosiddette joint-ventures, al molteplice fine di: meglio competere sui mercati internazionali, ridistribuire all'interno dell'azienda il portafoglio prodotti, rivedere la definizione di settore strategico nel modo più pragmatico e proficuo ai fini dell'eliminazione dei punti di perdita economica. Guerci presenta la sua proposta come alternativa alla privatizzazione secca delle imprese pubbliche, che a suo dire è oggi in Italia una via meno realistica e meno efficace

La proposta appare convincente e largamente condivisibile un po' meno per la verità lo è il fondamento su cui essa è basata. In ogni caso un utile contributo al dibattito, che l'importanza e l'attualità dell'argomento meritano, può venire dall'analisi dell'esperienza italiana di questi anni '80 e dall'individuazione delle difficoltà che le Partecipazioni statali hanno finora incontrato nel fare joint-venture serie con i privati

La conclusione cui si arriverà alla fine di questo contributo è che in realtà le joint-venture (pubblico privato) raramente decollano, spesso abortiscono,

TFG EF TPV

TFG EF TPV

Figure 3.7: The document classification view



# Chapter 4

## SAClet internals

### 4.1 Application structure and technologies

SAClet is a modern web application<sup>1</sup> which is immediately usable by users because it doesn't need to install anything. Designing the solution I chose modern web development technologies.

In figure 4.1 you can see the architecture schema of the solution composed by three main components:

- SAClet web application (front-end)
- SAClet core (back-end)
- SAClet task runner

I decided to split the project in three main modules that use a standard way to work together; each one defines an interface to interoperate with the others and in the future it would be possible to change every single part of the system without any technical constraint.

---

<sup>1</sup><https://github.com/martino/saclet>

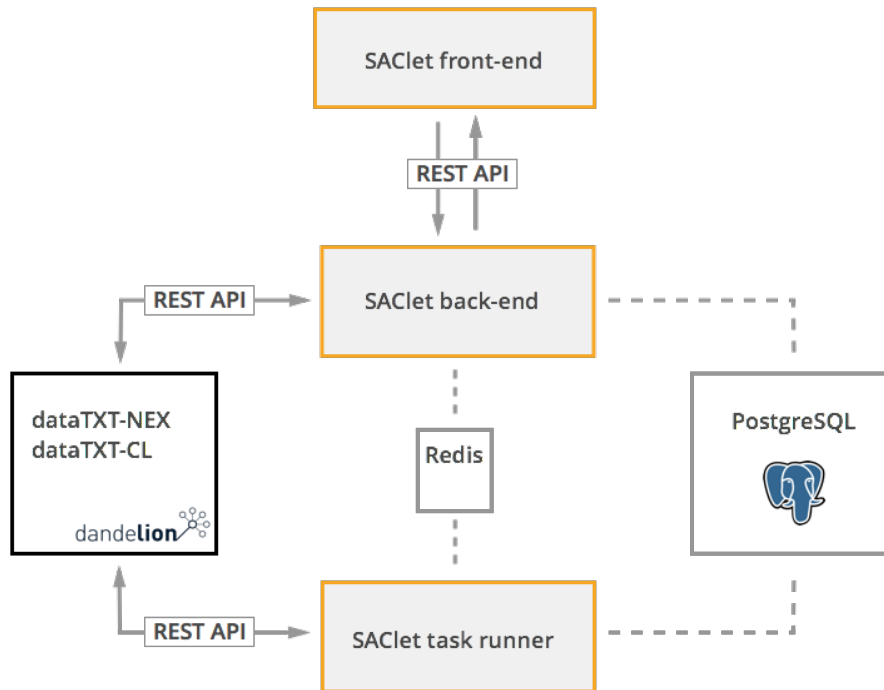


Figure 4.1: System architecture

## The web application

The front-end is a modern single page application built using AngularJS framework <sup>2</sup>; I choose it because it permits to create complex pages in an easy way; actually there are basically four reasons I am going to talk about to explain my choice.

First of all I would like to underline that bi-directional binding feature keeps always in sync user interfaces and data structures, thus saving a lot of code and common bugs.

Secondly I used *directive*<sup>3</sup>, a feature that permits to write custom HTML tag with a specific behavior; in my opinion, in the future this will be the way to build web applications composing them with a set of small reusable components (the same idea at the base of the LEGO).

<sup>2</sup><https://angularjs.org>

<sup>3</sup><https://docs.angularjs.org/guide/directive>

Thirdly, the other amazing feature that came out of the box is the testing framework; thanks to which you can write unit and functional tests to coverage every line of code that you write in order to avoid introducing bugs while developing new functionalities.

Last but not least there are already a lot of third part libraries that simplify the development process.

For the presentation layer I decided to use Bootstrap<sup>4</sup>, a CSS framework that provides out of the box a responsive grid and all common elements that you need developing a web application. Some of these components work with small javascript that are available as angular directives.

To build the application I choose *Yeoman*<sup>5</sup> as scaffolding tool; this is a “swiss knife” that allows to spare a lot of time to developing web applications. Its main functionality consists in providing a mechanism to handle each phase of the development, from the generation of the initial code skeleton to the creation of a final optimized application deployable on any web server.

Yeoman is basically a set of tasks that help developers to handle a lot of common problems, here below I will describe some of them.

First of all it provides a set of generators in order to scaffold the initial skeleton of any web application. For example there is a generator that creates the base html page using all best practices to ensure the maximum compatibility with all browsers while another generator initializes any kind of angularJS components.

Furthermore it uses bower to give developers a smart mechanism to include third part libraries handling versions requirements and conflicts.

During the development process it runs a web server with a live reload plugin that reloads the page in the browser when it detects any changes in the source code.

The final advantage is that it creates an optimized version of the web application minifying all resources (javascript, css, images), attaching a version

---

<sup>4</sup><http://getbootstrap.com>

<sup>5</sup><http://yeoman.io>

to each file to avoid problems with browsers cache and fixing all references in the source codes.

The deploy of the SAClet web application is trivial, you just need a web server that serves static files. This kind of deploy that doesn't need handling any kind of dependency is very comfortable.

## The back-end

The back-end structure is more complicated because it is built from various component that work together with the goal to provide data to the front-end and then to the users.

The principal components are the SAClet core and the task runner, the first one is a REST web service that talks with the front-end handling all synchronous requests; all time consuming tasks run in a dedicated the task runner.

The communication between the task runner and the web service uses distribute message passing; as messaging server I chose Redis<sup>6</sup>, a NOSQL database that can handle messaging queue. It isn't persistent but I think that it is the most rational solution in terms of maintenance costs.

The heart of the system is the SAClet core that is written in Python using Django web framework<sup>7</sup>. It saves all data into a relational database using a powerful ORM. There is not any constraint on the choice of the database server; on development machine it is very comfortable to use simpleDB, but on production machine I rather prefer using PostgreSQL server. The communication with the front-end uses restful API following the standard the facto way to expose services.

Long tasks are executed inside an asynchronous task runner that uses the same relational database with the SAClet core in order to share results with it. An example of tasks we must run asynchronously is the classification of a document set actually we cannot interrupt the site for a couple of minutes.

---

<sup>6</sup><http://redis.io>

<sup>7</sup><https://www.djangoproject.com>

The task runner that I used is Celery<sup>8</sup> which is written in python and works very well with Django.

## External services

The main external service that I used in SAClet is the Dandelion dataTXT APIs<sup>9</sup>; these provided me all the semantic text analysis I needed.

The most important one is dataTXT-CL<sup>10</sup>, the classifier that allowed me to store classifier models and to do the text classification returning a list of percentage of categories' coverage.

Each classification model is proxied by the SAClet core in order to provide a simple version management; as side effect we avoid to do a request to dataTXT for every changes users do in the classification model.

During the analysis of the users knowledge base I used dataTXT-NEX<sup>11</sup> in order to extract named entities needed by the automatic generation of classification models. In this process I also used a modified version of dataTXT-NEX that uses a page rank algorithm to extract only most significant named entities. This version works better on long texts because page rank permits to filter relevant entities dropping the ones which have no links with the others.

I used Dandelion wikisearch API<sup>12</sup> to help users to find the right topics and add them in a classifier model's category. It uses an index created on all wikipedia labels used to link each page in order to provide a smart autocomplete service.

---

<sup>8</sup><http://www.celeryproject.org>

<sup>9</sup><https://dandelion.eu/products/datatxt/>

<sup>10</sup><https://dandelion.eu/products/datatxt/cl/demo/>, which at the moment is available only with an invitation because it is in beta

<sup>11</sup><https://dandelion.eu/products/datatxt/nex/demo/>

<sup>12</sup><https://dandelion.eu/docs/api/datagraph/wikisearch/getting-started/>

### 4.1.1 Deploy

SAClet doesn't have any particular server requirement to be executed; you only need a machine with a web server, a python environment, Redis and an installation of PostgreSQL.

I chose Digital Ocean droplet<sup>13</sup> as the host machine where to install SAClet in order to save time in the machine basic configuration; also Amazon EC2 would be a valid solution but it is more expensive, the only reason to choose it would be scalability. In facts, the server that hosts the entire solution has 1 core processor, 1GB of memory and 30GB of SSD disk, nothing special.

On this machine runs an Ubuntu 14.10 but I've installed also an instance of PostgreSQL<sup>14</sup> (an object-relational database used as a store for the back-end), and an instance of Redis (a key-value store, used as a broker for the asynchronous task runner). The Django application runs into a Gunicorn WSGI server<sup>15</sup> and long tasks are handled by a Celery worker. To keep this asynchronous tasks under control I installed a web based tool named Flower which is very useful to understand the system performance.

To automatize the deploy process I used dploy.io<sup>16</sup>, a continuous deployment system that gets the source code from Github<sup>17</sup>, pushes it on the production server and restarts the updated services. I found it less powerful than other tools like Fabric<sup>18</sup>, Ansible<sup>19</sup> or Puppet<sup>20</sup> but I choose it because it does not need any kind of setup on the developer machine.

---

<sup>13</sup><https://www.digitalocean.com/features/technology/>

<sup>14</sup><http://www.postgresql.org>

<sup>15</sup><http://gunicorn.org>

<sup>16</sup><http://dploy.io>

<sup>17</sup><https://github.com>

<sup>18</sup><http://www.fabfile.org>

<sup>19</sup><http://www.ansible.com/home>

<sup>20</sup><https://puppetlabs.com>

### 4.1.2 Scalability

The system is designed to be scalable, it can handle the classification of large sets of documents without any problems. The key factor that ensures this scalability is that all heavy tasks run with celery workers designed to run on multiple machines. That way adding a new elaboration node is a simple operation; you just let this node to communicate with broker and database.

Figure 4.2 shows how the scaled system works.

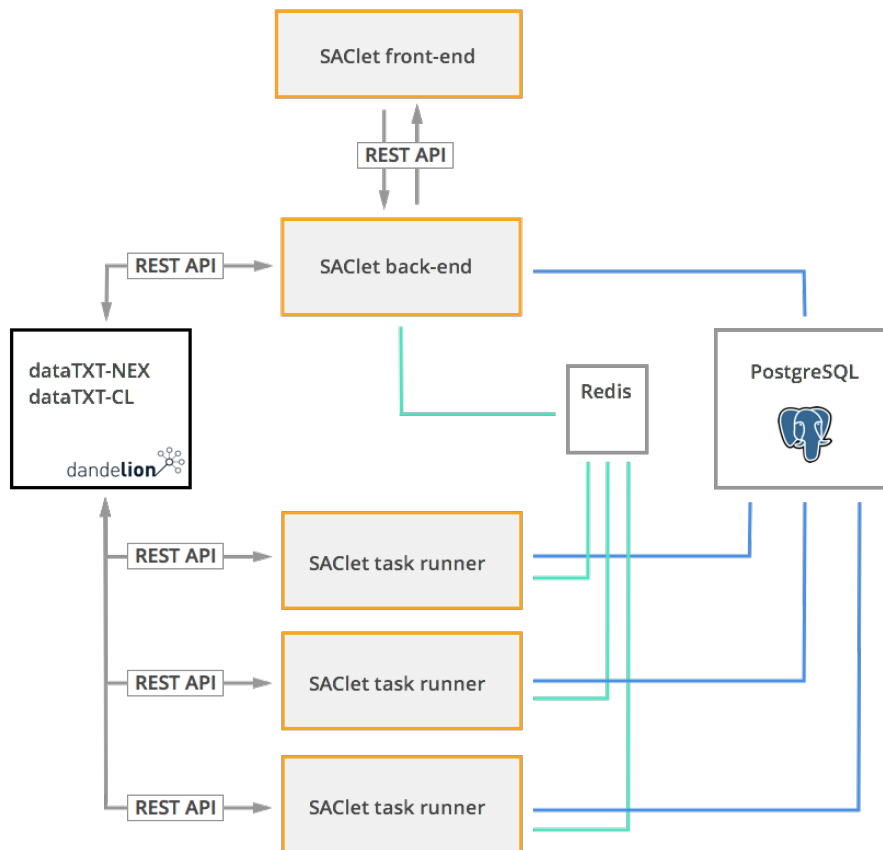


Figure 4.2: Scaled system architecture

## 4.2 Algorithms

The main goal in developing SAClet was the creation of a web application for users who do not want to know how modern classification technologies work but just need to use them to classify a lot of documents and discover hidden meanings.

The more demanding task was then to design and implement the entire process starting from the creation of a classification model to finally use it in the classification of document sets.

The most relevant algorithms that powered all the platform were developed into dataTXT API including the classification algorithm and the named entity extraction.

### 4.2.1 Classification models generation

The only interesting algorithm present into the project is the one which generates classification models from the users knowledge base.

The system is able to read and import as knowledge base the output results exported from nVivo; this is the most popular software that helps qualitative researchers to do qualitative data analysis. Actually I noticed that our users use this software to classify a large set of documents and produce a set of frames composed by a lot of nodes. For that reason I thought the system should be able to transform these frames into categories and for each one extract the most representative named entities. A graphical representation of this task is showed in figure 4.3.

So at the beginning the algorithm tries to extract named entities from the frames (I used dataTXT-NEX to accomplish this task without using any kind of NLP technologies); in this way I got a powerful tool able to contextualize short text fragments very well. For example, analyzing the phrase: “*Maradona scored a goal to Mexico*” using a plain NLP tool it is impossible to understand that Mexico stands for the football team rather than for the Country.



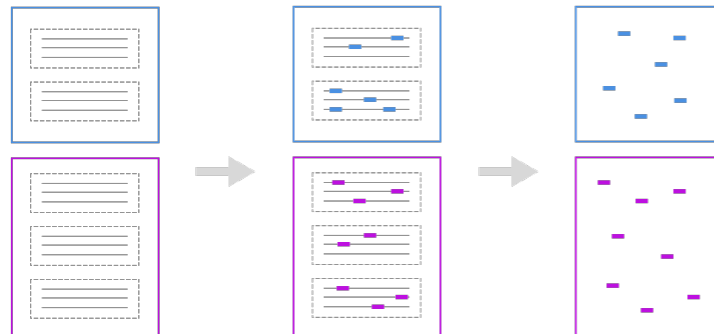


Figure 4.3: How system generates classification models

Using the semantic graph provided by Wikipedia, dataTXT can extract “Maradona”, “Goal” and “Mexico” as named entities <sup>21</sup>.

Another typical problem in this kind of tasks is polysemy, but again with a connected graph under the hood dataTXT can solve this problem in an easy way. Let us think about these two phrases: “the astronomer photographed the star” and “the paparazzi photographed the star”, it is impossible to distinguish the meaning of star using only the word context.

In that case DataTXT provides a powerful tool to extract this named entities; it works very well on short text yet on longer ones we can encounter problems if the texts switch context many times. In order to overcome this problem, I equipped the algorithm with an option that enables users to use a modified version of dataTXT-NEX using an algorithm inspired from the page rank to filter out only relevant entities.

At this point, for each nodes, that we call categories the system has collected a set of named entities (that we call topics). Now it has to select only the most representative ones; the length of selected sets is variable but we see in [VFS12] the best size is in a range between 15 and 30 and the algorithm lets users choose.

The results of the selection of topics that recur most frequently shows that some of them appear in many categories; this fact can be a confusion

<sup>21</sup>This analysis is here available: <http://bit.ly/17DUk6q>

factor in the classification phases.

To solve this problem I implemented a sort of *tf-idf* schema [RU11] to keep the most representative topics in each category using the formula 4.1 where  $freq(t, c)$  is the frequency of the topic  $t$  in the category  $c$ ;  $C$  is the number of all categories and  $C(t)$  is the number of categories containing  $t$

$$rank_c(t) = freq(t, c) * \log \frac{|C|}{C(t)} \quad (4.1)$$

The first part of the formula considers the relevance of the topic in each category and the second one is useful to reduce the relevance in ratio with the number of categories in which it appears.

The output of the algorithm is a set of categories that are mapped one to one with input frames that contain a variable number of representative topics; this is the classification model used by dataTXT-CL to classify any texts.

# Chapter 5

## Evaluation

### 5.1 Classification models

I started the whole process with qualitative research results and a powerful semantic text classifier; in order to create a new powerful system I wanted to combine them.

The qualitative research was based on the work of some researchers who had found about 600 articles written from 1/1/1984 to 31/12/1995 and published on *La Repubblica* and *Il Sole 24 Ore* concerning the privatization of Italian public steel industries. Once they had read them several times they extracted about 2000 relevant pieces of texts called references. Then they grouped references into nodes that represented key concepts (for example: privatization leads to unemployment). These nodes were then grouped into sets called frames that became the categories of our classification model.

The development process started with some experiments on this classification; the main goal was to create a classifier model that could be used both as a tool to test the work researchers did by hand and to classify any set of documents.

At the very beginning I started playing with input data just to experiment how to train a classifier system; I did not use any scientific methods to judge the results.

Researchers gave me different sets of frames but to better understand the results of my work I choose to use the one you can see in table 5.1 as reference.

Frame
Neo Liberalism (NL)
Technical Pragmatic View (TPV)
European Forces (EF)
Stigma Inertia “Boiardi” (SI)
Stigma Lottizzazione (SL)
Captious View Privatization (CSVP)
Continental Capitalism (CC)
Developmental State (DS)
Protecting Rents (PR)
Labour versus Capital (LVSC)
Taken for Granted (TFG)
Capture View Privatization (CVP)

Table 5.1: First set of frame

The first experimented idea was to start from the entire documents from which references were extracted. For each frame I analyzed all articles extracting named entities with dataTXT from which I was also returned the confidence score accorded to each match.

I created topics grouping the matches of the same entity so that the relative confidence was the sum of the related entities’ confidence. In order to retain a subset containing only the most relevant topics I chose to keep the ones with the highest confidence.

The idea was that for each category the system should identify the most relevant topics according to their confidence; in every category the weight of topics was normalized in a scale from 1 to 10.

The result was a little bit crappy because, when I tried to classify the

entire document set I noticed that only few documents were classified in the right way furthermore all documents were classified in the same way.

Looking at the classifier model I found that some topics appeared in many categories, for example *privatizzazioni* appears in all frames, nevertheless it is interesting to underline that most important thing is not the fact that it has been used but the way it has been used in the analyzed texts. For that reason I decided to proceed removing these topics. Unluckily the results were unsatisfactory because the classifier was not able to classify a big amount of documents.

I thought that each category had a too much restricted set of topics representing it so I tried to enlarge the category coverage in an automatic way. To do this I used dataTXT-REL that uses the WLM measure [WM08] to extract the ten most relevant related entities for each topic; these new topics were inserted into each category whose weight was calculated multiplying the the weight of the original topic with the measure of relatedness. Then I filtered again the topics using their weight to drop the less relevant ones in each category, nevertheless results were poor. Even in qualitative research it is not bad to proceed by trial and error.

Then I realized that using the confidence that dataTXT returns as way to prioritize each topic was a mistake because the importance cannot be defined using the estimation of the quality of the annotation; so I decided to use the occurrence number of every entity as weight for each topic. The classification model built with this method seemed to be better but I noticed that I had removed a lot of significant topics only because they appeared in many categories.

Before I changed again the algorithm to generate the classification model I needed a pragmatic way to understand if every new implementation produced a better result. I started to build up a system to this evaluation using recall, precision and f-score as parameters to judge every model and to better understand the results I computed a macro and a micro version of each one.

Finally I did not need to understand in an empirical way how each new

model performed nevertheless the first evaluation results were unacceptable, my classification model had been judged with an f-score of 0.1.

So I decided to modify the algorithm to use a method inspired from *td-idf* schema to avoid the deleting of common topics in each category giving them an appropriated weight. The intuition was right, the new classification model scored a 0.33 f-score, better than previous one but it was poor too.

Then I started to review the entire process, and I decided to start from the frames rather than from full articles. That way I used the researchers' work to extract topics from the frames; some of them were very short but with dataTXT-NEX this was not a problem because it performs very well also with small context.

Another problem that I had to face was the use of dataTXT-REL actually all the new topics added in this way did not help in any way the classifier because during the classification process each category is expanded in the same way.

I also decided to use a more scientific method to test my classification model, using different sets of nodes to create the classification model and test it.

I started using the same set mainly for two reasons: on the one hand first results were very bad and I wanted to use all knowledge to build better models, on the other hand some sets of frames were composed by few nodes as you can see in table 5.1.

For each frame I selected the same number of nodes to extract the set of relevant topics.

The resulting classification models scored an f-score of 0.49 which was better but not amazing.

Actually the output of every test were only composed by numeric values so that it was impossible to understand the performance of each category and it was really hard to create better models. I improved my testing platform introducing confusion matrix, a tool that visually let users understand how each category performed. An example is visible in figure 5.1, you can see

Frames	Number of references
Neo Liberalism — NL	498
Technical Pragmatic View — TPV	381
Continental Capitalism — CC	253
Stigma Inertia “Boiardi” — SI	166
Stigma Lottizzazione — SL	121
Developmental State — DS	89
European Forces — EF	89
Protecting Rents — PR	78
Labour versus Capital — LVSC	60
Taken for Granted — TFG	58
Capture View Privatization — CVP	46
Captious View Privatization — CSVP	28

Table 5.2: Size of each frame

that NL and LVSC hide all the other categories.

These results were very encouraging because in the meanwhile researchers found the same problem with the frame set (some categories were too much weak). It was the first proof that the overall system worked in the right way.

The knowledge base was stored into a relational database and thanks to this changing the frame set grouping was really trivial, you can find the new frame set described in table 5.1.

The new classification model performed better scoring an f-score of 0.53; the main problem, as we can see from the confusion matrix, was that some categories couldn't be discovered from the classifier. This was because they had a too much small amount of nodes in the frame and the system wasn't able to extract a good set of relevant topics to represent it.

I changed the way to select the set of nodes used to extract the most relevant topics, choosing for each frame half the nodes. It improved again the classification model so that the final one got an f-score of 0.59 without

		TEST OUTCOME										
	PR	NL	CC	TFG	SI SL CSVP	CVP	LVSC	TPV EF	DS	empty	total	
<b>PR</b>	5%	34%	5%	-	5%	-	31%	2%	10%	5%	38	
<b>NL</b>	-	67%	4%	0.4%	0.8%	-	7%	14%	2%	3%	244	
<b>CC</b>	-	67%	6%	-	0.9%	-	7%	8%	3%	6%	111	
<b>TFG</b>	-	75%	6%	-	3%	-	3%	3%	-	6%	29	
<b>SI SL CSVP</b>	-	65%	1%	-	2%	-	7%	9%	7%	5%	158	
<b>CVP</b>	-	30%	-	-	-	-	34%	21%	13%	-	23	
<b>LVSC</b>	-	13%	3%	-	-	-	76%	3%	-	3%	30	
<b>TPV EF</b>	-	53%	3%	-	-	-	6%	23%	7%	5%	226	
<b>DS</b>	-	51%	4%	-	-	-	20%	15%	6%	2%	45	
<b>empty</b>	-	-	-	-	-	-	-	-	-	-	0	

Figure 5.1: Confusion matrix

any kind of human action.

I really appreciated the fact that to improve the generation of classification model I used the tools developed to test it with profit. I found them very useful to understand how each classification model category performs and how each topic influences the overall performance. These results give users a lot of hints on how to change the classification models avoiding wrong choices. Also the minimal version control for the models was very precious because it let users change categories without any problems; if they did wrong changes they could easily rollback on the last good version.

Only a researcher that knows the overall context can understand what important topics are missing in every category and can refine the classification model.



Frame
Neo Liberalism (NL)
Technical Pragmatic View (TPV)
European Forces (EF)
Taken for Granted (TFG)
Stigma Inertia “Boiardi” (SI)
Stigma Lottizzazione (SL)
Captious View Privatization (CSVP)
Continental Capitalism (CC)
Developmental State (DS)
Protecting Rents (PR)
Labour versus Capital (LVSC)
Capture View Privatization (CVP)

Table 5.3: Second set of frame

## 5.2 Documents classification

The final step of SAClet’s evaluation is the analysis of the outcomes of automatic text classification.

To classify each document the system splits it into paragraphs, then it classifies them and combines the results in order to create an overall score. In this way it is able to find all categories contained in each text.

Afterwards, to compare the automated classification results, I used the manual classification researchers exported from nVivo as goal standard. Results are visible in table 5.2 and reflect the evaluation of the classification models clearly showing the fact that some categories are completely hidden by others. Some other categories like *Protecting Rents* (PR) and *Developmental State* (DS) were not very well recognized because they are a sort of shade categories.

The performance of the classification is influenced by two factors:

- the knowledge base used by the system to generate the classification model is too small and for some frames it hasn't got enough references to extract a significant list of topics. The only solution could be using human knowledge to add the missing topics to every scarcely covered category;
- all texts were written in Italian, so dataTXT used had to use a knowledge base built on the Italian Wikipedia as reference even though it is not as rich as the English one.

In the meanwhile I tried to use MALLET on the same article set concerning the Italian steel privatization. I have found a lot of alternative classifications that researchers consider interesting, but none of them really represented their original ideas. For this reason it was impossible for me to compare the two classification systems.

run	PR	NL	CC	LVSC	CVP	SI	SL	CVSP	TFG	EF	TPV	DS
1	0	27	0		22			12		38		0
2	0	29	0		30			17		44		0
3	0	37	0		46			34		60		1
4	0	49	0		57			50		71		7
5	0	56	0		57			52		70		1
6	0	60	1		61			60		75		2
7	4	63	3		38			18		80		0

Table 5.4: Right percentage of document classification

# Chapter 6

## Conclusions

The output of my thesis is a web application called SAClet enabling users to take advantage of the most recent text classification technologies based on text semantics to analyze large sets of documents.

The aim of my work is to enhance the workflow of scholars using frame analysis as a tool for their researches. In fact, the system is able to create a classification model based on the work researchers had previously done. That way users can upload any set of documents, analyze it with the classification model previously created and check the validity of their hypothesis on a new corpus of documents. This process permits scholars to avoid the manual classifications of every new set of documents; as a result they are able to process even the largest ones in an efficient, quick automated way.

Furthermore the system allows them to modify their classification model in order to improve it or experiment with new ideas. Without an evaluation system it is very hard to understand how model changes impact on the documents classification; SAClet provides tools which are able to judge classification models and enable users to become aware of their performance.

The most precious one is the confusion matrix, an instrument that provides a magnifying glass to better understand the behavior of each class. In this way users can visually see whether there are wide classes masking the others or too small ones. During my work this tool has been able to find

some issues in the classification models that in the meanwhile, researchers had manually detected.

The system permits also users to create a classification model starting from a simple intuition with no further input; this way SAClet does not provide any evaluating tools.

Nevertheless the fact that Italian Wikipedia is not as complete as the English one influenced the performance of system. Actually the English version of Wikipedia is richer in both contents and the links between them, and for that reason it is closer to the human knowledge. This has a great impact both on the construction of the classification model and on the final text classification.

Compared to MALLET, SAClet allowed both professor Edoardo Mollona and dr. Luca Pareschi to import their research about steel privatization in Italy into the system; then they could improve it by grouping similar categories, and finally apply it on new big corpora of texts.

I think that this tool represents an improvement for the workflow of qualitative researchers; in facts thanks to this, they have a platform to evaluate their manual work and to explore new sets of documents. With some extra work we are leveraging this platform to the next level.

Now I am going to discuss current ideas that concern the evolution of this product.

First of all I would like to highlight that, on the generated classifier model, I have seen that some topics could belong to many categories and the discriminating factor is the polarity of the context phrase. With this in mind I thought that users could set rules for each topic. So doing, during the classification phase, the system will discover the polarity of a text and, as a consequence, it will build the right classification model with the right topics. The sentiment analysis for each phrase could well be done with some services like Alchemy<sup>1</sup> that provides an API to fill in the classification pipeline without too much work. I am sure that this feature can significantly improve

---

<sup>1</sup><http://www.alchemyapi.com/products/alchemylanguage/sentiment-analysis>

classification performances.

Referring to the intuition that each topic could have some rules, we will add a temporal validity for each one. In this way the system would try to classify a text by inserting it into a temporal frame and building a suitable classification method. If the system is not able to frame a text it could obtain the information it needs from the metadata of the document itself.

Another feature we are developing is the visualization of classification results [Ble12], which enhances the current navigation by category. Since the system extracts named entities to classify documents, we will reuse them to create new navigation patterns based on co-occurrences of entities (like persons and places) and of categories. We will also consider categorizing a set of documents using multiple classification models and use all the results as powerful facets filters.

A further interesting filter that we are implementing is based on links between named entities. For instance starting from the entity of an Italian prime minister, the system will let users filter out any documents concerning all his predecessors. We are developing it using the interesting project of *Linked Data Fragments* [VHDM<sup>+</sup>14] on *DBpedia*<sup>2</sup>.

In addition to this, we will propose for users a set of classification models from any kind of documents sets in order to give researchers some hints on the documents' hidden meanings. As a matter of fact, we will use the power of MALLET to do this and find a way to import its classification models into SAClet. However to make this possible are searching for a way to connect the words contained in each category to Wikipedia topics.

Moreover we can state that dataTXT is able to classify any web page, thus we will create streams of URL from any sources like a twitter stream or a newspaper RSS and classify them in real time. In this way scholars could analyze events at the very moment they occur.

Furthermore SAClet could increase its power by using an improved version of dataTXT. This should be able to use all Wikipedia versions to build a

---

<sup>2</sup><http://fragments.dbpedia.org>

single knowledge base containing pages and links in any languages. With this enhancement SAClet could be able to analyze the Italian documents using classification models built on an English knowledge base.

In conclusion I can thus summarize the main goals I have achieved with SAClet: first of all it allows scholars to enhance their research based on frame analysis; secondly it permits to evaluate and improve any classification models in a smart way and thirdly it enables researchers to classify any set of documents using their previous research again.

# Bibliography

- [ANH<sup>+</sup>68] David Paul Ausubel, Joseph Donald Novak, Helen Hanesian, et al. Educational psychology: A cognitive view. 1968.
- [BGJT04] David M Blei, Thomas L Griffiths, Michael I Jordan, and Joshua B Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:17, 2004.
- [Ble12] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.
- [BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- [DNB13] Paul DiMaggio, Manish Nag, and David Blei. Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of us government arts funding. *Poetics*, 41(6):570–606, 2013.
- [FS10] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.
- [Gof74] Erving Goffman. *Frame analysis: An essay on the organization of experience*. Harvard University Press, 1974.

- [LM06] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM, 2006.
- [Mat09] Jörg Matthes. What’s in a frame? a content analysis of media framing studies in the world’s leading communication journals, 1990-2005. *Journalism & Mass Communication Quarterly*, 86(2):349–367, 2009.
- [McC02] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
- [MH10] Renate E Meyer and Markus A Höllerer. Meaning structures in a contested issue field: A topographic map of shareholder value in austria. *Academy of Management Journal*, 53(6):1241–1262, 2010.
- [RU11] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.
- [VFS12] Daniele Vitale, Paolo Ferragina, and Ugo Scaiella. Classification of short texts by deploying topical annotations. In *Advances in Information Retrieval*, pages 376–387. Springer, 2012.
- [VHDM<sup>+</sup>14] Ruben Verborgh, Olaf Hartig, Ben De Meester, Gerald Haendonck, Laurens De Vocht, Miel Vander Sande, Richard Cyganiak, Pieter Colpaert, Erik Mannens, and Rik Van de Walle. Querying datasets on the web with high availability. In *The Semantic Web–ISWC 2014*, pages 180–196. Springer, 2014.
- [WD08] Pu Wang and Carlotta Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–721. ACM, 2008.



- [WM08] Ian H. Witten and David Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, AAAI Press, Chicago, USA, pages 25–30, 2008.

# Acknowledgements

The only reason I am able to graduate now is because I had special people who encouraged and supported me in this adventure.

I would therefore like to start by thanking professor Fabio Vitali, my supervisor, for having introduced me to the world of the semantic web and for having dedicated his time to follow my work.

I am also thankful to professor Edoardo Mollona and dott. Angelo Di Iorio for all of their assistance. I am specially thankful to dott. Luca Pareschi for having introduced me to social sciences, and for having followed me closely, at every step of the way.

I thank Michele Barbera, CEO of SpazioDati, for all of his precious hints in those decisive moments.

I thank all of my colleagues at SpazioDati, who have bravely resisted the temptation to kill me for so many days, while developing amazing software in the meantime.

I thank my parents who guided me from the sky.

I thank all of my relatives who helped me starting this challenge; all my friends who I have never forgotten, even in this demanding period.

And last, but not least, I thank my wife Francesca - for her skill at falling asleep close to me when I had to work late hours, and for her willingness to stay up and help me during the most critical moments.