istat
working
papers

N.19
2016

La nuova applicazione di codifica web dell'ATECO 2007: WITCH, un web service basato sul sistema di codifica CIRCE

Manuela Murgia, Valeria Prigiobbe

Istat
Istituto Nazionale
di Statistica

istat
working
papers

N.19
2016

La nuova applicazione di codifica web dell'ATECO 2007: WITCH, un web service basato sul sistema di codifica CIRCE

Manuela Murgia, Valeria Prigiobbe

Istat

Istat Working Papers

La nuova applicazione di codifica web dell'ATECO 2007: WITCH, un web service basato sul sistema di codifica CIRCE

N. 19/2016

ISBN 978-88-458-1914-8

La nuova applicazione di codifica web dell'ATECO 2007: WITCH, un web service basato sul sistema di codifica CIRCE[1]

Manuela Murgia, Valeria Prigiobbe

Sommario

Il working paper descrive l'attività svolta in Istat per la creazione di un web service dedicato alla codifica automatica dell'Attività economica Ateco 2007 per la quale sono state ridisegnate l'architettura e realizzato un nuovo software di codifica automatica. Sulla base dell'esperienza maturata, sono proposti progetti di ricerca finalizzati all'utilizzo di questo strumento in altri contesti di codifica riguardanti possibili sviluppi dell'algoritmo di codifica e possibili implementazioni di web service che offrano servizi diversificati in funzione sia delle caratteristiche della variabili da codificare sia del contesto in cui il servizio viene utilizzato e dei relativi metodi di codifica. Sono inoltre delineati i possibili vantaggi che la fase di raccolta dati dei processi di indagine potrebbe ottenere dall'adozione di tali strumenti.

Parole chiave: codifica automatica, web service, Ateco.

Abstract

The working paper describes the activity carried out in Istat to create a web service for the automated coding of the Economic Activity (NACE 2007) that required the re-design of the IT architecture and the development of a new software for textual variables coding. On the basis of this experience, research projects are proposed with the aim of spreading out the use of the web services in other production contexts. In particular, research projects should be focused on the development of software and web services able to offer a customised "coding services". Customisation should take into account variables' characteristics, coding methods and possible uses of the results.
Possible advantages that the data collection step of a survey process could obtain from the adoption of such instruments are also outlined.

Keywords: automatic coding, web service, ATECO

[1] Gli articoli pubblicati impegnano esclusivamente gli Autori, le opinioni espresse non implicano alcuna responsabilità da parte dell'Istat. I paragrafi sono a cura di: Manuela Murgia e Valeria Prigiobbe (1); Laura Capparucci (dal 2.1 al 2.6); Massimiliano Degortes (2.7); Loredana Mazza (dal 3.1 al 3.3); Angelina Ferrillo (3.4); Massimo Perri e Valeria Prigiobbe (4); Marco Pericò (5.1); Fausto Panicali (5.2); Paolo Di Domenico (6.2 e 6.3); Roberta Roncati (6.1 e 6.4); Diego Zardetto (7.1.1 e 7.1.2); Manuela Murgia (7.1, 7.2, 7.2.2 e 7.2.4); Manuela Murgia e Massimo Perri (7.2.1); Massimo Perri (7.2.3). In riferimento al tema trattato nel presente IWP, per WITCH si intende Web service Interface To Coding Handler, invece per CIRCE il Comprehensive Istat R Coding Environment.

Indice

1. Introduzione

Il presente *working paper* descrive l'attività svolta in Istat per la creazione di un web service dedicato alla codifica automatica dell'Attività economica – Ateco 2007 – e propone diversi filoni di ricerca finalizzati all'utilizzo di questo strumento in altri contesti di codifica.

Il progetto nasce della necessità di sostituire il prodotto esistente con un sistema basato sulle nuove tecnologie informatiche e compatibili con le piattaforme software utilizzate in Istituto.

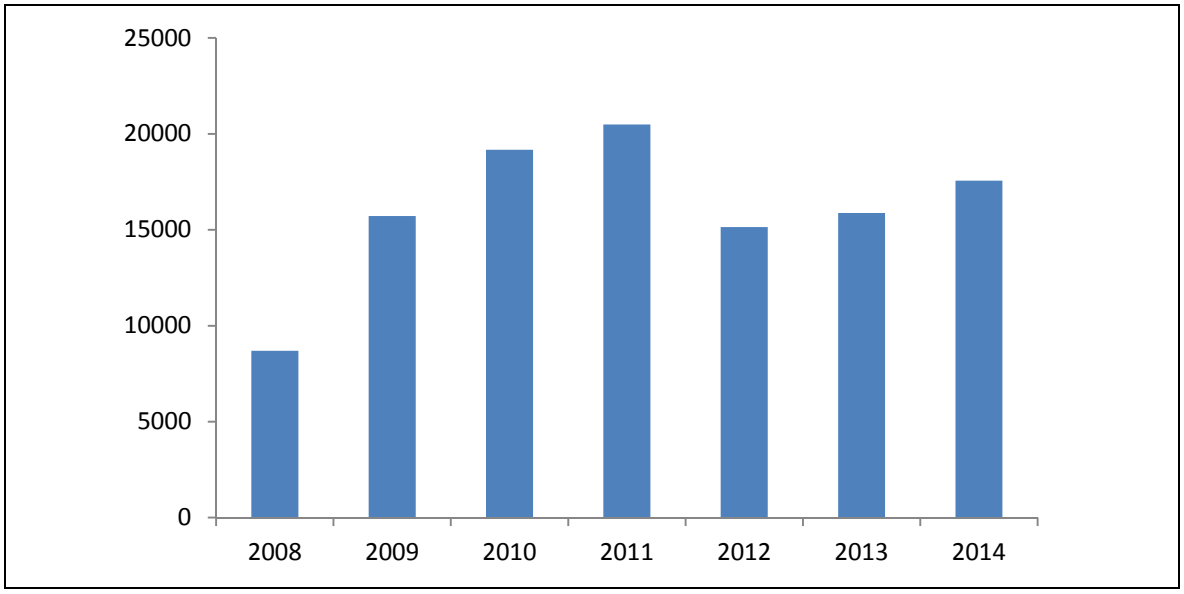
Per meglio comprendere la portata dell'innovazione introdotta, si riporta nel seguito una breve storia del sistema Istat per la codifica dell'Ateco.

Nel corso del 2007, in occasione del rilascio della nuova classificazione Ateco, l'Istat realizzò un'applicazione web, esposta sul sito dell'Istituto, per la consultazione della classificazione stessa e con il fine di far fronte alla sempre più crescente diffusione ed utilizzazione dell'Ateco da parte degli utenti esterni. Questo strumento, all'epoca, doveva essere di supporto sia agli Enti (Agenzia delle Entrate, Inail, Inps e Camere di Commercio) che dovevano adottare la nuova classificazione, sia ai cittadini per diversi usi amministrativi.

Da allora l'applicazione web è stata utilizzata da migliaia di cittadini ed istituzioni che, collegandosi al sito dell'Istituto, possono ricercare la propria attività economica ed aver indietro il codice relativo.

Dal 2008 il numero degli accessi al sito è stato altissimo, con medie settimanali che hanno raggiunto picchi di 20.000 accessi per attestarsi sui 15.000 (Allegato 1), come mostra il grafico seguente.

Figura 1 – Media settimanale delle *query* degli utenti del Sistema di codifica web dell'Ateco dal 2008 al 2014



In questi anni la base informativa sottostante il sistema di codifica è stata costantemente aggiornata valutando la capacità del sistema di codificare un insieme significativo di testi costituti dalle stringhe di ricerca dell'attività economica digitate dagli utenti web (*query utente*). Questo ha permesso di ottenere un'applicazione di codifica molto efficiente e di elevata qualità, come descritto nel capitolo 3.

Il "cuore" del sistema di codifica, che ha portato ai risultati sopra descritti, è stato, fino alla prima metà del 2015, il software ACTR v3, prodotto e distribuito da Statistics Canada e che l'Istat ha adottato anche per la codifica automatica di altre variabili quali la Professione, il Titolo di studio, il

Comune e Stato estero. Agli inizi del 2014 si è reso necessario progettare la sostituzione di ACTR v3 con un altro prodotto che fosse compatibile con le versioni Windows 7, Windows Server 2008 e superiori verso cui l'Istat stava migrando. ACTR v3, infatti, non era più funzionante sulle nuove piattaforme software non essendo, tra l'altro, più manutenuto dall'Istituto canadese[2].

È stato quindi costituito un gruppo di lavoro con l'obiettivo di migrare l'applicazione di codifica dell'Ateco 2007 verso un nuovo sistema che doveva:

1. essere del tutto trasparente per gli utenti finali che, quindi, non avrebbero dovuto percepire alcun cambiamento del sistema di codifica sottostante l'applicazione;
2. essere basato su un software per la codifica di variabili testuali che garantisse almeno gli stessi livelli di qualità della codifica raggiunti con il precedente sistema;
3. garantire livelli di sicurezza e prestazionali (numero di accessi contemporanei supportati e tempi di risposta per la codifica di una stringa) adeguati alle esigenze degli utenti e agli standard dell'Istituto.

Il gruppo di lavoro interdipartimentale è stato organizzato in modo tale da avere al suo interno colleghi appartenenti a strutture esperte nella realizzazione di ogni singola attività. In particolare:

- progettazione, sviluppo ed ottimizzazione del nuovo software di codifica: Laura Capparucci, Maria Teresa Buglielli e Diego Zardetto;
- progettazione ed implementazione dell'interfaccia grafica e creazione della base informativa e dei file di standardizzazione: Massimiliano Degortes e Loredana Mazza;
- controllo della qualità della codifica: Angelina Ferrillo e Alberto Valery;
- progettazione dell'architettura web e del backend application/web service: Massimo Perri, Marco Pericò e Fausto Panicali;
- implementazione del backend application/web service: Massimo Perri;
- attività di test di sicurezza e di carico: Marco Pericò e Fausto Panicali;
- supporto sistemistico: Fausto Panicali.

L'attività del gruppo è stata coordinata da Valeria Prigiobbe e Manuela Murgia.

I capitoli che seguono descrivono in dettaglio l'attività svolta. Il capitolo 2 riguarda il nuovo pacchetto di codifica, che si basa sul software R e che è stato denominato **CIRCE** – **C**omprehensive **I**stat **R** **C**oding **E**nvironment.

Il capitolo 3 descrive i passaggi principali per la realizzazione dell'ambiente di codifica in CIRCE e la procedura di valutazione del nuovo software in termini di efficienza e qualità. Il capitolo 4 riguarda l'applicazione di codifica web WITCH - **W**eb service **I**nterface **T**o **C**oding **H**andler, della quale CIRCE è il motore della codifica. Importante anticipare l'aspetto innovativo dell'applicazione che è stata totalmente riprogettata nell'ottica di realizzare un web service migliorando, rispetto al passato, la funzionalità, la sicurezza, l'affidabilità, la portabilità e la riusabilità.

Il capitolo 5 descrive i test di sicurezza e di carico necessari al rilascio di un'applicazione sicura rispetto ad attacchi esterni e robusta in previsione di un elevato numero di accessi contemporanei.

Il capitolo 6 illustra il nuovo strumento per la consultazione, navigazione e codifica della classificazione Ateco, disponibile sul sito web istituzionale, realizzato per accogliere l'applicazione di codifica web WITCH e orientato al rispetto degli standard di accessibilità.

Infine, il capitolo 7 è dedicato agli sviluppi futuri relativi sia ai miglioramenti del pacchetto di codifica che alla possibilità di estendere il servizio web ad altre tipologie di classificazioni e modalità di codifica, in modo da rendere più fruibile l'utilizzo di questa classe di strumenti da parte dell'utenza interna ed esterna all'Istituto.

[2] ACTR v3 è stato sostituito da Statistics Canada con un altro prodotto, che, dopo essere stato testato ed analizzato, si è dimostrato non adatto a soddisfare le esigenze di codifica dell'Istat.

2. Il nuovo pacchetto di codifica CIRCE

Il nuovo pacchetto di codifica, che ha sostituito ACTR v3 e si basa sul software R, ha come scopo l'attribuzione automatica di un codice a partire da un testo, indipendentemente dalla lingua e dalla classificazione adottata. Finora tale funzione era stata assicurata utilizzando il software ACTR v3, sviluppato da Statistics Canada, ma attualmente non più supportato.

ACTR v3, inoltre, non era più compatibile con i nuovi sistemi Windows 7 e Windows Server 2008, pertanto è stato necessario sviluppare un nuovo software di codifica che garantisse gli stessi risultati quali-quantitativi di ACTR v3. Vista la necessità di migrare l'applicazione web esposta sul sito dell'Istituto per la consultazione e codifica dell'attività economica, si è deciso di applicare prioritariamente il nuovo pacchetto alla codifica web della variabile Ateco.

Oltre all'urgenza della migrazione al nuovo pacchetto, necessaria a garantire continuità al servizio di codifica web, nello sviluppo del nuovo strumento di codifica bisognava tenere presente che ACTR v3 veniva utilizzato anche in ambiente pc, sia per la ricerca del codice riferito a singole stringhe, che per la codifica *batch* di interi file. Inoltre, era utilizzato anche per la codifica di altre variabili afferenti a classificazioni ufficiali (Professione, Titolo di studio, Comune e Stato estero).

La sviluppo in R ha reso CIRCE portabile su diversi ambienti senza necessità di compilazione. Questo ha permesso di realizzare un unico pacchetto di codifica funzionante sia in ambiente Windows che Linux. Quindi, a differenza di ACTR v3, CIRCE è utilizzabile sia in ambiente pc, attraverso un'interfaccia grafica utente, come descritto nel paragrafo 2.7, che in ambiente web, attraverso la "chiamata" ad un web service.[3]

Attualmente è previsto l'accesso al web service dedicato alla codifica web dell'Ateco e accessibile tramite la pagina del sito istituzionale[4]. Il servizio potrà essere replicato per la codifica delle altre variabili testuali afferenti alle classificazioni ufficiali.

Data la necessità di mantenere una continuità con il passato, il nuovo pacchetto riproduce il comportamento di ACTR v3, ma, essendo sviluppato in casa, offre l'opportunità di modifiche e/o aggiunte di nuove funzionalità (capitolo 7) .

Di contro, occorre dire che CIRCE (essendo basato su R) non ha la stessa velocità di esecuzione propria dei pacchetti scritti utilizzando linguaggi compilati, come nel caso di ACTR v3. Tale criticità è particolarmente sentita per la funzione di codifica *batch*, i cui tempi sono superiori rispetto al precedente software. Per quanto sia già stata fatta un'ottimizzazione del codice che ha permesso di diminuire notevolmente i tempi di esecuzione, questi risultano ancora troppo lunghi. È in fase di studio la possibilità di sviluppare un contesto che lavori in modalità multi-processore per rendere più veloce l'applicazione (capitolo 7).

Con R, inoltre, le funzioni realizzate, raccolte in un package, possono eventualmente essere inserite nel repository del CRAN per essere utilizzate dalla comunità scientifico/statistica internazionale.

2.1. Funzionamento del software

CIRCE, come ACTR v3, prevede un confronto tra la stringa di testo da codificare e le voci contenute nel dizionario della classificazione.

Sia il dizionario che il testo da codificare vengono preliminarmente sottoposti ad un processo di standardizzazione o *parsing* che ha lo scopo di eliminare dal testo la variabilità grammaticale o sintattica che non incide sull'aspetto semantico, ma soltanto sulla forma, e che pertanto è irrilevante ai fini dell'abbinamento con le voci del dizionario.

La prima operazione da eseguire è quindi quella del caricamento della base informativa relativa allo specifico dizionario della classificazione. In questa fase vengono anche calcolati i "pesi" delle singole parole che fanno parte del dizionario.

[3] Per ulteriori dettagli sul funzionamento del software è possibile consultare il Manuale Utente disponibile sul sito Istat http://www.istat.it/it/strumenti/metodi-e-strumenti-it/strumenti-di-elaborazione/circe

[4] Il web service per la codifica dell'Ateco è accessibile attraverso al pagina http://www.istat.it/it/strumenti/definizioni-e-classificazioni/ateco-2007

Nella fase di codifica vengono selezionate le voci del dizionario che risultano essere uguali o simili al testo da codificare in base alle parole che hanno in comune. La selezione avviene sulla base del punteggio di *matching* che viene calcolato considerando sia i pesi delle parole che alcuni parametri che stabiliscono le soglie di precisione. Nel caso in cui la corrispondenza sia esatta, oppure venga selezionata una sola voce che supera un certo valore soglia, viene associato al testo da codificare un codice univoco o "unico". Nel caso in cui la corrispondenza non sia esatta, o non superi una certa soglia, la procedura fornisce un output composto dalle voci del dizionario che più si avvicinano al testo ("multipli" e "possibili"). Se non c'è alcuna voce del dizionario che supera la soglia minima, il testo da codificare viene definito "fallito" (per la descrizione dell'algoritmo di *matching* si rimanda al paragrafo 2.5).

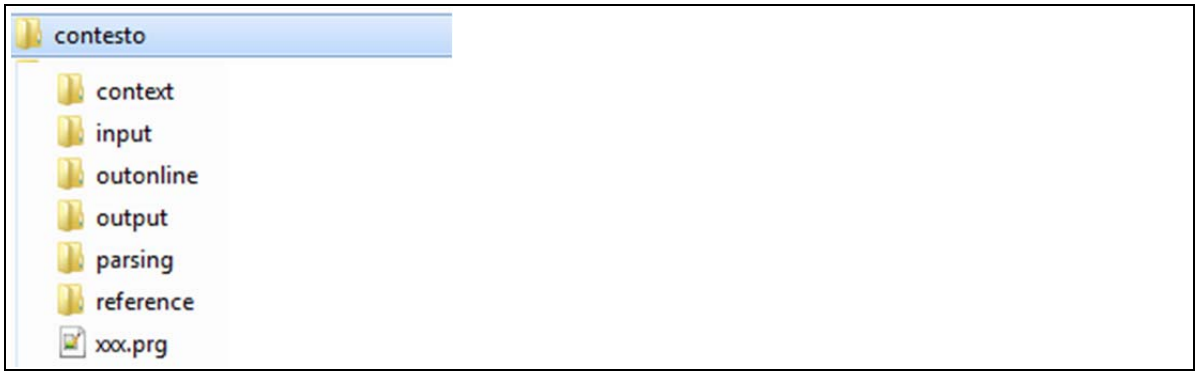
2.2. Il contesto

Sia sulla piattaforma Linux che Windows, CIRCE prevede una cartella in cui è definito il "contesto", cioè l'ambiente in cui viene eseguita la codifica.

Il "contesto" rappresenta l'insieme di tutti i file che concorrono a definire la base informativa necessaria per la codifica, in relazione alla classificazione adottata (Ateco, Professione, ecc.). Pertanto il contesto è costituito da: il dizionario della classificazione di riferimento, la strategia ed i file di *parsing*, il file di input (ovvero i testi da codificare), i file di output contenenti i risultati della codifica ed il file di progetto.

La struttura della cartella di contesto è rappresentata nella figura sottostante.

Figura 2 - Struttura della cartella contesto



Le singole sotto-cartelle contengono:

- *context*: il dizionario standardizzato della classificazione di riferimento, i file relativi ai pesi delle parole ed il file degli eventuali duplicati, l'ambiente R contenente il database;
- *input*: il file di input per la codifica *batch*;
- *outonline*: il file di appoggio per la visualizzazione della codifica interattiva effettuata attraverso la maschera utente;
- *output*: i file di output della codifica *batch* ed il report sull'esito della codifica;
- *parsing*: i file predisposti per la standardizzazione dei testi ed il file della strategia di *parsing*;
- *reference*: il dizionario della classificazione di riferimento, ed il relativo tracciato.

Il file di progetto, genericamente indicato con "xxx .prg**",** contiene la descrizione della struttura del file di input per la codifica *batch* ed i parametri della strategia di codifica, così come sarà descritto nel paragrafo 2.7.

2.3. Standardizzazione del testo

La fase di standardizzazione di CIRCE, chiamata anche *parsing*, ricalca quella di ACTR v3, ampiamente descritta nel volume *"Metodi e Software per la codifica automatica e assistita dei dati",* Tecniche e strumenti n.4 -2007, ed utilizzata come base di partenza.

È stato tuttavia necessario mettere a punto una fase di sperimentazione empirica in quanto non si disponeva né del codice sorgente di ACTR v3 né di una descrizione esaustiva e tecnicamente dettagliata del suo funzionamento. Procedendo empiricamente è stato possibile avere informazioni fondamentali sulla procedure di trasformazione dei testi e di calcolo del punteggio di *matching*.

Sia nella fase di caricamento del dizionario che in quella di codifica, tutti i testi vengono standardizzati attraverso una serie di trasformazioni (*parsing*) realizzate in sequenza. Il *parsing* è costituito da diverse funzioni di trasformazione, quali, ad esempio, la rimozione dei caratteri ininfluenti, delle parole inutili, di suffissi/prefissi, l'individuazione di sinonimi, ecc. ed ha l'obiettivo di rimuovere tutte le varianti grammaticali e sintattiche in modo da rendere uguali due descrizioni diverse ma dallo stesso contenuto semantico.

Il *parsing* è logicamente suddiviso in quattro fasi principali:

1. pre-trattamento;
2. trattamento delle stringhe;
3. trattamento delle parole;
4. post-elaborazione.

Il **pre-trattamento** è orientato ad eliminare e/o sostituire tutti i caratteri che sono ininfluenti o potrebbero creare ambiguità nei successivi passaggi (es. spazi iniziali e finali, vocali accentate, ecc.).

Nel **trattamento delle stringhe** il testo è elaborato come una stringa continua di caratteri. Il principale obiettivo di questa fase è di facilitare il riconoscimento di particolari sequenze, così come appaiono nel testo da codificare. Risulta particolarmente utile per la gestione delle abbreviazioni standard (es. T.V.) e per la gestione di alcune particolari sequenze di caratteri.

Dopo il trattamento delle stringhe il testo viene suddiviso in parole, che costituiranno quindi l'input delle fasi successive. Il **trattamento delle parole** consiste in sostituzioni e/o eliminazioni delle stesse in base alle specifiche indicate nelle trasformazioni previste e nella gestione di prefissi, suffissi e caratteri doppi. Infine nella **post-elaborazione** avviene l'eliminazione delle parole duplicate e l'ordinamento alfabetico delle stesse.

Tutti i file relativi a questo processo si trovano nella sottocartella *parsing* del contesto: ad ogni file è associato un tipo di trasformazione. L'ordine con cui saranno effettuate le trasformazioni è indicato nel file *strategy.txt*.

A differenza di ACTR v3, CIRCE non prevede controlli né sulla sequenza delle trasformazioni, né sul numero di ripetizioni delle stesse. Questa scelta è stata fatta per superare le limitazioni di ACTR v3, offrendo così un maggiore grado di flessibilità e, quindi, di personalizzazione della strategia di codifica, anche in funzione di eventuali sviluppi futuri. E' importante precisare comunque che la personalizzazione della strategia deve avvenire rispettando la sequenza delle quattro fasi sopra descritta.

Nell'appendice B è riportata la tabella con il dettaglio delle possibili trasformazioni dei testi.

2.4. Calcolo dei pesi

Anche per questa fase è stato implementato un algoritmo analogo a quello di ACTR v3 e descritto in "*Metodi e Software per la codifica automatica e assistita dei dati",*Tecniche e strumenti n.4 -2007. Tuttavia alcune formule e le relative descrizioni si discostano leggermente da quanto scritto nel testo citato, in quanto sono il risultato delle prove empiriche effettuate per comprendere dettagliatamente il comportamento di ACTR v3.

Nella fase di caricamento del dizionario viene creata una tabella che associa ad ogni parola presente nel dizionario un peso, secondo la formula:

$$P(W_i) = 1 - \log(NW_i)/\log(N) \qquad (1)$$

dove NW_i è il numero di codici contenenti la parola i-esima (W_i), mentre N è il numero totale di codici contenuti nel dizionario.

Nel caso in cui nel dizionario siano presenti più voci con lo stesso codice, al fine del calcolo dei pesi, queste vengono preventivamente accorpate in una sola mantenendo tutte le parole univoche riferite alle singole voci. Attraverso la sperimentazione è stato possibile capire che il termine "numero dei codici", usato nella documentazione disponibile, doveva far riferimento ai soli codici univoci contenuti nel dizionario, tenendo però in considerazione tutte le parole diverse afferenti allo stesso codice.

I pesi così calcolati sono utilizzati nella fase di codifica per il calcolo del punteggio di *matching*.

2.5. Calcolo del punteggio di *matching* e codifica

Nella fase di codifica, il testo da codificare è confrontato con il dizionario per la ricerca di un abbinamento esatto (*direct match*), ossia la ricerca della singola voce del dizionario che abbia tutte le parole in comune con il testo da codificare. Se l'esito è positivo darà luogo, inequivocabilmente, all'assegnazione di un codice unico. Se, invece, il tentativo fallisce viene ricercato un abbinamento parziale (*indirect match*). In questo caso il *software* individua, tramite una misura della similarità tra testi (S) di tipo empirico, "il codice" o "i codici" del dizionario con descrizione più simile al testo da codificare. A tal fine, vengono selezionate tutte le voci del dizionario che hanno almeno una parola in comune con il testo da codificare ed alle quali viene assegnato un punteggio di *matching* che è funzione del numero di parole in comune e del loro grado d'informatività (peso) all'interno del dizionario di riferimento. Le voci selezionate vengono successivamente ordinate in ordine decrescente di punteggio.

La formula di calcolo del punteggio (S) è la seguente:

$$S = 10(a + 2b)/3 \quad (2)$$

dove

$$a = 2N_C/(N_R + N_D)$$
$$b = 2\sum_i P(W_i^C)/\{\sum_j P(W_j^R) + \sum_i P(W_i^D)\}$$

dove N_C è il numero di parole in comune tra il testo da codificare e la singola voce del dizionario, N_R e N_D rappresentano il numero totale di parole contenute, rispettivamente, nel testo da codificare e nella singola voce del dizionario, mentre $P(W_i^C)$, $P(W_J^R)$ e $P(W_i^D)$ rappresentano i pesi definiti nella (1), rispettivamente della i-esima parola in comune (W_i^C), della j-esima parola contenuta nel testo da codificare (W_J^R) e della i-esima parola della singola voce del dizionario.

La misura di similarità espressa nelle (2) assume valori compresi nell'intervallo [0,10] i cui estremi corrispondono ad un abbinamento testuale nullo (S=0) o ad un abbinamento esatto (S=10).

La regione di accettazione per la misura di similarità è data dalle relazioni (3) ed è costruita utilizzando tre parametri soglia, Smin, Smax e ΔS, che rappresentano, rispettivamente, le soglie minima e massima di accettazione e la distanza minima tra il punteggio massimo (S1) ed il successivo (S2) attributi alle voci del dizionario con cui è stato realizzato il *match*. A seconda del dove si "collocherà" il punteggio di *matching* rispetto alla regione di accettazione, si avranno diversi risultati della codifica, suddivisibili in i codici Unici, Multipli, Possibili o Falliti. In maggior dettaglio:

- UNICO: S_1>Smax e {[(S_1- S_2)>ΔS] o S_2 n.e. o S_2 < Smax} (3a)[5]

[5] Si precisa che la formalizzazione della (3a) è diversa rispetto a quella contenuta in "*Metodi e Software per la codifica automatica e assistita dei dati"(Tecniche e strumenti n4-2007)* in quanto esplicita un numero maggiore di casistiche possibili comprendendo anche i casi in cui S_2 non esiste oppure ha un punteggio inferiore alla soglia minima.

- MULTIPLI: $S_1 > Smax$ e $(S_1 - S_2) \leq \Delta S$ (3b)
- POSSIBILI: $Smin < S_1 \leq Smax$ (3c)
- FALLITI: $(S_1 \leq Smin)$ o S_1 n.e. (3d)

Se è soddisfatta la condizione (3a) la voce del dizionario con punteggio massimo (S_1) è dichiarata "vincente", il codice che le è associato è unico e viene assegnato in modo completamente automatico al testo di input. I casi descritti nelle (3b) e (3c) necessitano, invece, di un'analisi finalizzata a valutare la presenza di un codice corretto tra quelli proposti dal sistema da assegnare al testo di input.

I valori dei parametri soglia, Smin, Smax e ΔS, sono fissati dall'utente in funzione degli obiettivi di codifica. Valori alti aumentano la precisione della codifica, ossia la percentuale dei codici unici corretti, a scapito però del tasso di codifica, ossia della percentuale di codici unici assegnati. Il viceversa accade per valori bassi dei parametri di soglia. Quindi la scelta dei valori "ottimali" da attribuire ai parametri sarà il frutto di prove di codifica finalizzate ad individuare quei valori che permettano un bilanciamento tra quantità di testi codificati univocamente (*recall rate*) e qualità degli stessi (*precision rate*).

2.6. Output della codifica

La struttura ed il contenuto dell'output di una procedura di codifica dipendono, ovviamente, dal tipo di codifica effettuata:

- per la codifica interattiva il risultato viene visualizzato nella maschera utente come descritto nel paragrafo 2.7;
- per la codifica web viene prodotto un output in formato json;
- per la codifica *batch* i risultati vengono scritti nella sotto-cartella "output" del contesto e sono costituiti da un report che contiene una sintesi sull'esito della codifica e da diversi file che contengono le relative tipologie di *match* (il contenuto di detti file è descritto nel paragrafo 2.7).

È importante precisare che la precedente applicazione web per la codifica dell'Ateco non era un'applicazione web nativa, ma un adattamento dell'applicazione di codifica *batch*. Infatti presentava due punti critici:

1. la scrittura su file residenti sul server;
2. la gestione delle richieste utente in modalità sequenziale.

Queste problematiche sono state risolte da CIRCE, che ha reso quindi possibile la creazione di un web service per la codifica, come descritto nel capitolo 4.

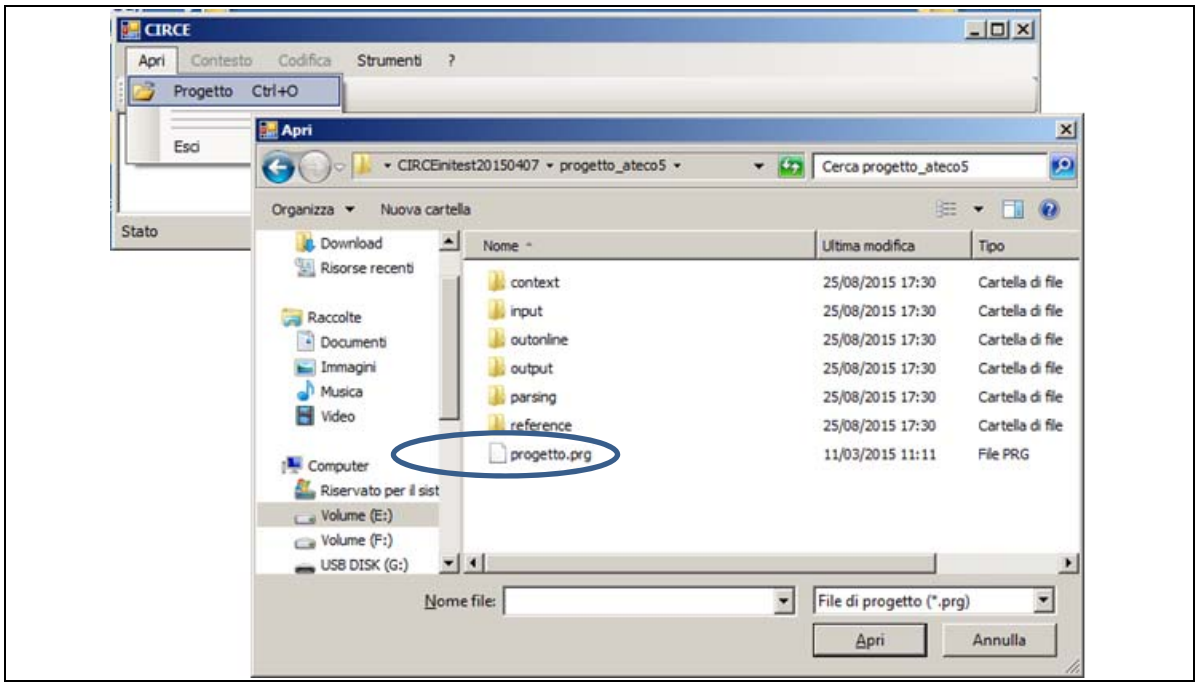
2.7. L'interfaccia utente

Per l'uso di CIRCE in ambiente pc (Windows) è stata realizzata una interfaccia grafica - Graphical User Interface (GUI) - che permette di eseguire un progetto di codifica *batch* e di gestire il *matching* interattivo di singole stringhe rispetto ad un determinato contesto. In particolare la GUI permette di:

1. selezionare un progetto di codifica;
2. modificare un progetto se già esistente;
3. creare il contesto di codifica;
4. gestire la codifica interattiva o *batch*;
5. scegliere una delle strategie di codifica definite all'interno del file di progetto.

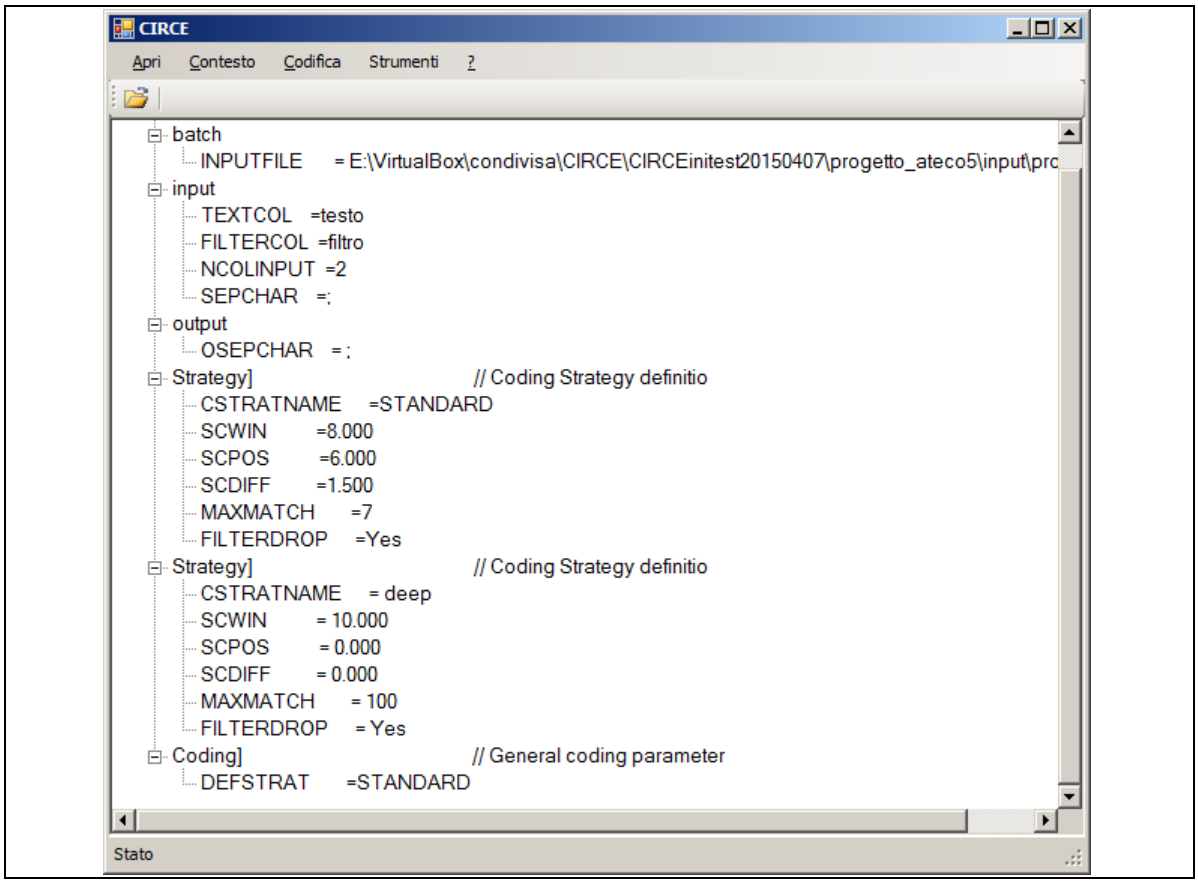
Il file di progetto (Appendice A) che contiene la descrizione della struttura del file di input per la codifica *batch* ed i parametri delle strategie di codifica può essere selezionato o modificato attraverso la GUI come mostrato nelle figure seguenti.

Figura 3 - Scelta del progetto



Il progetto selezionato verrà visualizzato nella schermata principale.

Figura 4 - Contenuto del file di progetto



Nel progetto sono specificate le seguenti informazione:

- dove si trova il file di input da codificare (INPUTFILE);
- la sua struttura (input);
- le diverse strategie di codifica con i relativi parametri (Strategy) e quale tra queste si intende utilizzare (Coding).

Dopo la definizione del file di progetto è possibile creare un contesto di codifica, ossia predisporre il dizionario per la fase di *match*. Il file di testo relativo al dizionario viene caricato nel sistema e sottoposto alla fase di standardizzazione secondo le regole definite dalla strategia di *parsing*.

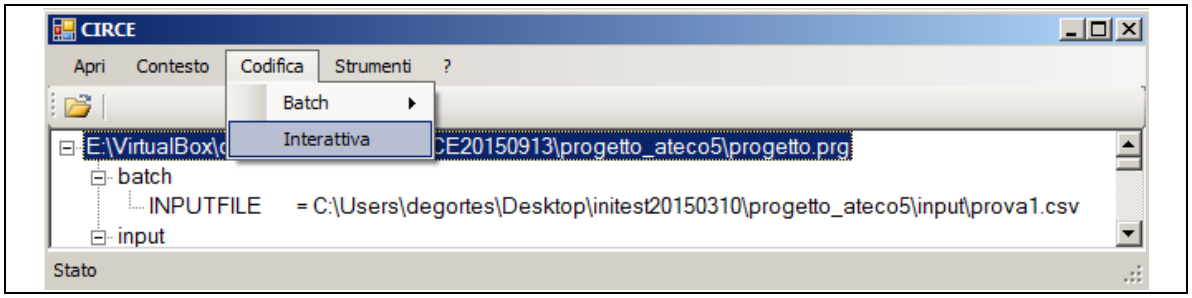
I risultati di questa fase sono rappresentati dal dizionario standardizzato, il relativo "database", i file relativi ai pesi associati alle singole parole (paragrafo 2.2) ed agli eventuali duplicati. Questi ultimi due file sono direttamente consultabili dall'utente, in quanto:

- il file dei pesi permette di capire "l'influenza" delle singole parole sulla fase di *match* e rappresenta quindi un ausilio all'ottimizzazione delle regole di *parsing* e, di conseguenza, dei risultati della codifica;
- il file dei duplicati contiene quei record che possono essere o duplicati effettivi (stesso codice e stessa standardizzazione) oppure vere e proprie incompatibilità (stessa standardizzazione, ma codice diverso) che debbono essere analizzate e risolte in quanto in loro presenza CIRCE non consente la creazione del contesto.

Dopo la creazione del contesto, può iniziare la fase di codifica, che può essere di due tipi: interattiva o *batch*. Di seguito una dimostrazione del funzionamento della GUI per la realizzazione dei due tipi di codifica.

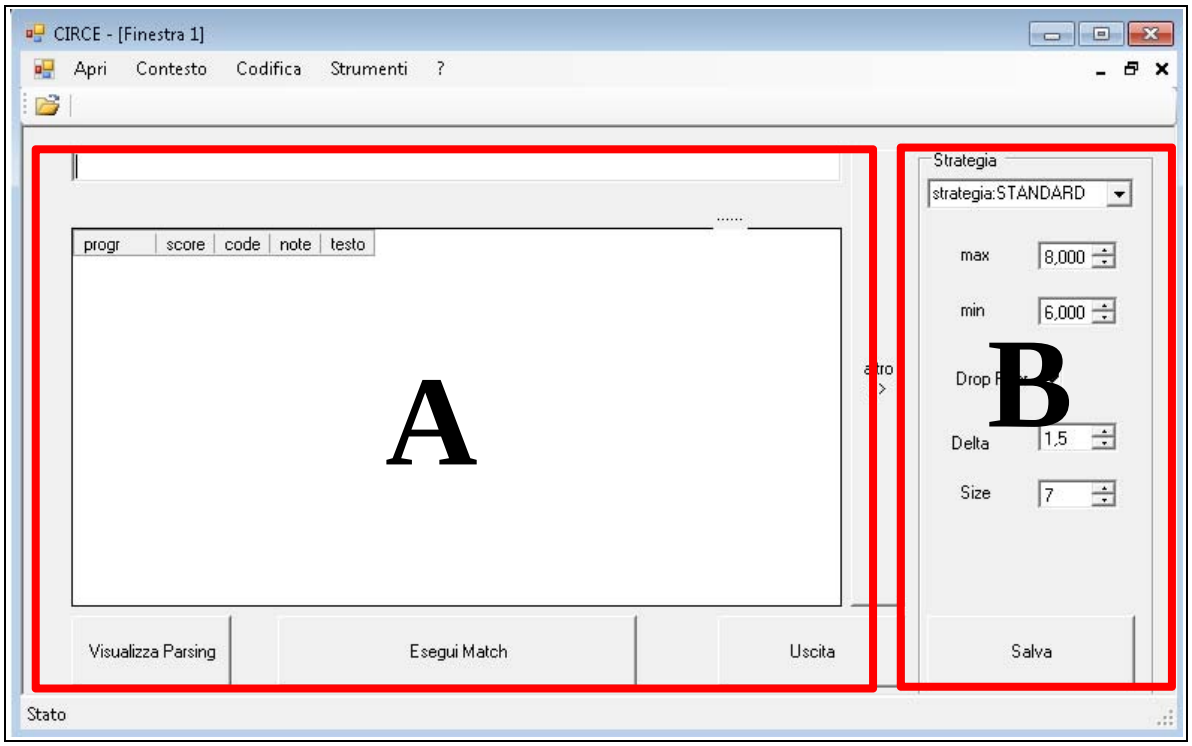
Selezionando il pulsante "Codifica" è possibile eseguire la codifica secondo la modalità desiderata.

Figura 5 - Selezione tipologia di codifica



Scegliendo "Interattiva" apparirà la seguente schermata, che può essere divisa in due parti, contraddistinte da A e B, come nella figura seguente.

Figura 6 - Schermata della codifica interattiva



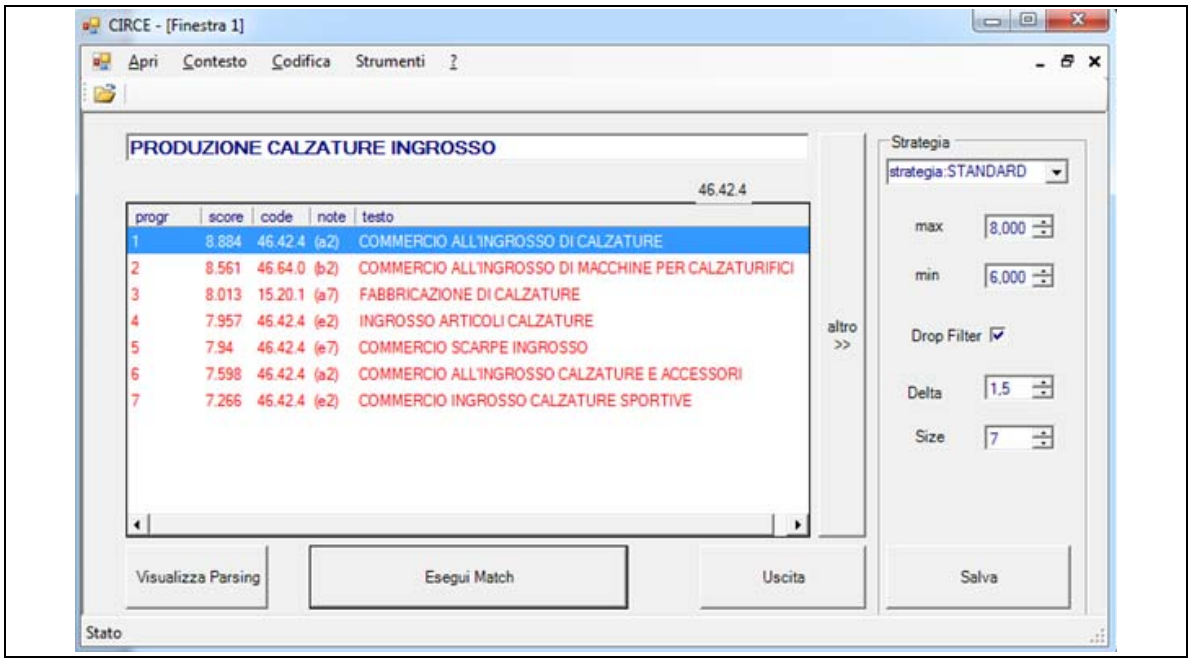
Nella parte A si inserisce il testo da codificare, digitandolo nell'apposito box in alto. Nella parte B è possibile configurare alcuni parametri (paragrafo 2.5) relativi alla strategia di codifica: si può selezionare la strategia, tra quelle definite nel file di progetto, e si possono modificare i valori dei parametri relativi ai punteggi di *match* (max, min e Delta), all'uso del filtro (Drop Filter)[6] ed al numero di risultati che si vuole siano mostrati a video (Size).

Con "Esegui *Match*" viene eseguita la procedura di codifica e visualizzato il risultato del *match*.

Come esempio nella figura seguente è stato riportato il risultato del *match* del testo "PRODUZIONE CALZATURE INGROSSO" che presenta un elevato livello di similarità con diversi testi presenti nel dizionario. Poiché i punteggi del *match* sono poco distanziati tra loro e compresi tra il punteggio massimo ed il minimo, il risultato della codifica rientra nella casistica dei "Multipli".

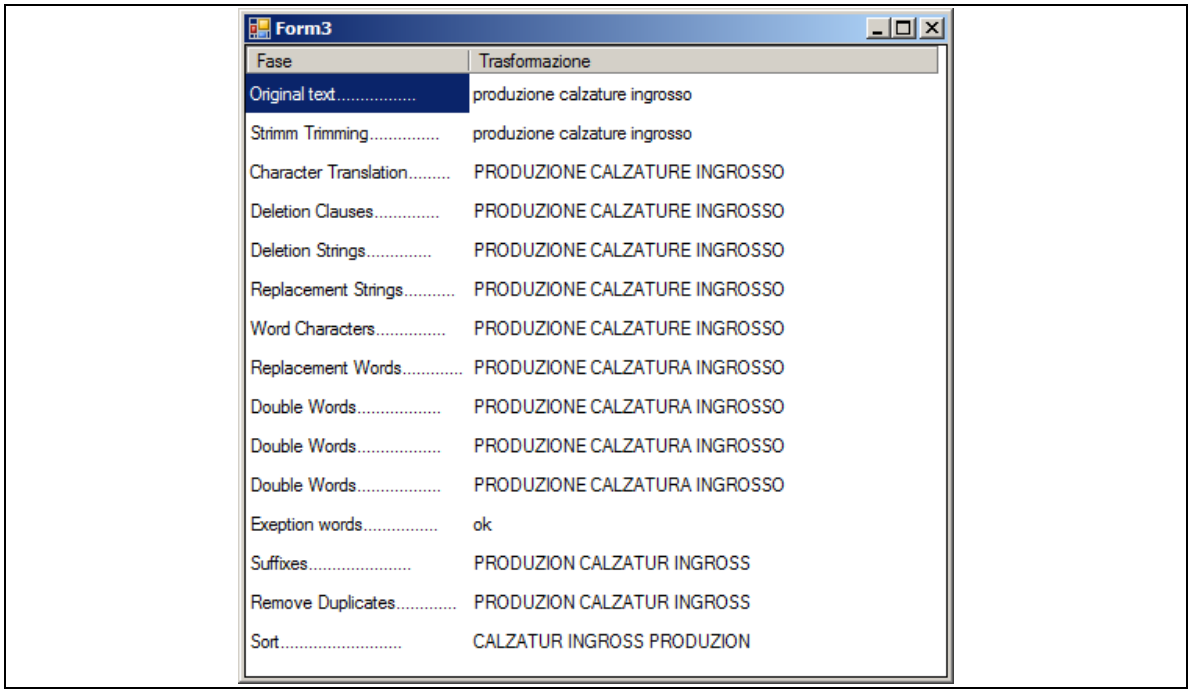
[6] La funzione che gestisce l'uso del filtro è stata predisposta, ma sarà implementata nel prossimo futuro.

Figura 7 - Risultato della codifica interattiva



Con la funzione "Visualizza *Parsing*" è possibile verificare come il *parsing* abbia elaborato il testo di input (Figura 2.7).

Figura 8 - Risultati dei vari passi del parsing



L'analisi delle trasformazioni che subisce il testo di input ad ogni passaggio della strategia di *parsing* è fondamentale nella fase di addestramento dell'ambiente di codifica.

Selezionando "*Codifica > Batch"*, dopo aver scelto la strategia (Figura 2.8), sarà possibile eseguire la codifica secondo questa modalità.

Figura 9 - Codifica Batch

Il file da codificare è specificato all'interno del file di progetto alla voce "INPUTFILE". Prima di procedere alla codifica, viene eseguito un passaggio di "pulizia" del file in cui vengono eliminati dei caratteri diversi da valori alfanumerici, utilizzando un'espressione regolare per effettuare le sostituzioni all'interno dei file da codificare.

Tra i file di output è prodotto anche un report che sintetizza i principali risultati della codifica, come il numero dei *match* unici diretti (unici con massimo punteggio di *match*), che rappresenta un indicatore di qualità della codifica (Tavola 1).

Tavola 1 – Esempio di report di codifica

Tipo di codifica	Valori assoluti	Valori %
Unici	6.690	44,49
Multipli	932	6,20
Possibili	5.794	38,54
Falliti	1.619	10,77
Totale	**15.035**	**100,00**
Numero di Direct *Match*	5.249	

I file dei risultati permettono di capire con quale stringa o stringhe del dizionario sono state abbinate le descrizioni di input e quale sia il punteggio del *match*. In particolare, i file contengono: il progressivo della stringa da codificare e la stringa stessa, il tipo di *match*, il punteggio del *match*, il codice/i con cui è stata abbinata la stringa di input con associata la relativa descrizione contenuta nel dizionario, l'origine della voce del dizionario indicata nel campo "Note" (ad esempio voce ufficiale o voce "empirica" derivante dal linguaggio dei rispondenti).

È possibile personalizzare il contenuto dei file indicando nel file di progetto quale altre informazioni del file di input si vogliono esportare nel file di output.

Di seguito sono riportati, a titolo di esempio, il contenuto di un file di "Unici" e di "Multipli".

Tavola 2 – Esempio di contenuto file Unici

Progr	Stringa di input		Tipo	Score	Codice	Note	Descrizione dizionario
1	SERVIZI DEI CENTRI PER IL BENESSERE FISICO	*	U	10	96.04.1	(b1)	SERVIZI DEI CENTRI PER IL BENESSERE FISICO
2	COLTIVAZIONE DI UVA	*	U	10	01.21.0	(ep)	AZIENDA AGRICOLA PRODUZIONE DI UVA
7	COMMERCIO AL DETTAGLIO AMBULANTE DI PRODOTTI ITTICI	*	U	10	47.81.0	(ei)	COMMERCIO AMBULANTE PRODOTTI ITTICI
64	ABBIGLIAMENTO PRODUZIONE	*	U	10	14.13.1	(e*)	PRODUZIONE ABBIGLIAMENTO UOMO E DONNA

Tavola 3 – Esempio di contenuto file Multipli

Progr	Stringa di input		Tipo	Score	Codice	Note	Descrizione dizionario
14	PRESTAZIONE DI SERVIZI AMMINISTRATIVI E CONTABILI	*	M	8,5	69.20.1	(ei)	PRESTAZIONE SERVIZI CONTABILI
14	PRESTAZIONE DI SERVIZI AMMINISTRATIVI E CONTABILI	*	M	8,5	69.20.1	(ei)	SERVIZI AMMINISTRATIVI CONTABILI
15	PRESTAZIONE DI SERVIZI AMMINISTRATIVI E CONTABILI	*	M	8,5	69.20.1	(ei)	PRESTAZIONE SERVIZI CONTABILI
15	PRESTAZIONE DI SERVIZI AMMINISTRATIVI E CONTABILI	*	M	8,5	69.20.1	(ei)	SERVIZI AMMINISTRATIVI CONTABILI
60	ABBIGLIAMENTO BAMBINI	*	M	9	14.13.1	(e*)	PRODUZIONE ABITI PER BAMBINI
60	ABBIGLIAMENTO BAMBINI	*	M	8,4	46.42.1	(e2)	COMMERCIO ALL'INGROSSO DI ABBIGLIAMENTO PER BAMBINO

Tra gli output della codifica *batch* si trova anche il file dei casi "falliti", ossia dei casi in cui il punteggio di *matching* non ha superato la soglia minima. In ACTR v3 questo file conteneva solo le stringhe di input mentre CIRCE evidenzia anche il codice che ha ottenuto il punteggio di *matching* massimo (pur non superando, ovviamente, la soglia minima). Questo tipo di output non solo è di ausilio alla fase di addestramento del sistema, ma può essere usato anche per la valutazione dei risultati della codifica nei casi in cui si mettano a confronto gli esiti della procedura automatizzata con quelli della codifica manuale.

2.7.1. Strumenti di supporto alla codifica

In questa parte della GUI sono riportate delle funzioni di supporto al processo di codifica che permettono di eliminare delle possibili fonti di errore. In particolare le funzioni offerte sono:

- pulizia del file di input per la codifica *batch*;
- creazione file in formato csv.

La funzione di pulizia del file di input permette di poter eseguire una "pulizia" preventiva del file da sottoporre a codifica *batch*, evitando problemi di interpretazione di alcuni caratteri, come quelli accentati, da parte di R. I problemi sorgono dal fatto che il sistema adotta la codifica UTF8, non totalmente compatibile con R, ma necessaria per la pubblicazione dei risultati su web.

La funzione per la creazione di file in formato CSV permette di selezionare un file di testo contenente delle semplici stringhe e di crearne uno in formato csv, elaborabile dalla procedura di codifica. Questa funzionalità risulta molto utile in quanto estende la codifica *batch* a set di stringe fornite in vari formati.

3. La codifica dell'ATECO: da ACTR v3 a CIRCE

3.1 La preparazione dell'ambiente di codifica

Le principali fasi di lavorazione previste da un qualunque sistema di codifica, sia automatica che assistita, includono la costruzione di un dizionario informatizzato e la standardizzazione delle descrizioni testuali in esso contenute. La base per la costruzione del dizionario è il manuale ufficiale della classificazione di riferimento, opportunamente trattato, in modo tale da inserire nel dizionario processabile solo descrizioni sintetiche, analitiche e non ambigue. I sistemi di codifica automatica prevedono, prima della codifica vera e propria, un'elaborazione dei testi attraverso la quale le descrizioni testuali vengono ridotte ad una forma "canonica" (standardizzazione) dipendente sia dalla lingua che dallo specifico dominio dell'applicazione.

Per la codifica automatica dell'attività economica (Classificazione ufficiale delle attività economiche Ateco 2007) è stata implementata, nel corso degli anni, una base informativa molto ricca, costituita da un dizionario informatizzato di 34.743 descrizioni associate ai codici della classificazione ufficiale (pari a 918 Categorie) e dai file di standardizzazione (*parsing)* per un totale di 17.370 sinonimi.[7]

[7] Somma delle trasformazioni contenute nei file di *parsing* relativi a stringhe di testo, parole singole o coppie di parole .

La migrazione della base informativa da ACTR v3 verso il nuovo sistema CIRCE è stata realizzata attraverso continui confronti tra i due prodotti finalizzati a raggiungere un duplice obiettivo:

1. ottenere la stessa standardizzazione dei testi;
2. ottenere la stessa efficacia (*recall rate*), ossia la percentuale di codici assegnati automaticamente sul totale dei testi da codificare, nonché la stessa accuratezza dei risultati (*precision rate*), ossia la percentuale di codici corretti sul totale dei testi codificati.

3.2 La fase di standardizzazione

La prima fase di lavoro è consistita nell'analizzare i vari passi di standardizzazione dei testi, secondo la strategia di *parsing* propria di ACTR v3 e di confrontare poi i risultati con quelli ottenuti con CIRCE per evidenziare eventuali divergenze o anomalie.

Il confronto tra i due processi di standardizzazione è avvenuto sulla base del dizionario informatizzato, utilizzato per la codifica dell'attività economica (Ateco 2007), che prevede al suo interno solo Categorie di Attività economiche associate a codici completi a 5 digit, costituito ad oggi da 34.067 record, e dai file di standardizzazione (17.370 sinonimi).

Il confronto è stato realizzato in step successivi, seguendo le fasi del processo di standardizzazione, proprie di ACTR v3 (implementate anche in CIRCE):

- pre-trattamento (trattamento dei caratteri);
- trattamento delle stringhe;
- trattamento delle parole;
- post-elaborazione (ulteriore trattamento delle parole).

Il primo step del confronto ha riguardato la fase orientata al riconoscimento dei caratteri e al trattamento delle stringhe; successivamente si è passati ad inserire gradualmente i vari processi orientati alle parole fino ad arrivare alla strategia completa così come prevista in ACTR v3.

Procedendo empiricamente, ed individuato l'ordine con cui ACTR v3 effettua le trasformazioni (analizza il testo di input da sinistra a destra ed esegue la prima trasformazione che trova), è stato possibile scrivere le regole di *parsing*, all'interno di ciascun file, nello stesso ordine usato da ACTR v3.

Per agevolare l'utente nella creazione dei file di *parsing* è stata importata in CIRCE l'*utility* (Appendice C) sviluppata per ACTR v3 che permette di controllare la coerenza interna ad ogni file e tra un file e l'altro. Tale software di supporto alla gestione del sistema è di fondamentale importanza, soprattutto nel caso di classificazioni complesse, come l'Ateco, dove i file di *parsing* contengono non solo sinonimi, ma anche i criteri classificatori. La buona riuscita della creazione del contesto di riferimento dipende, infatti, dai controlli sui file di standardizzazione, che devono essere effettuati ogni qualvolta vengano inserite nuove trasformazioni, sia relative a semplici sinonimi che a regole classificatorie.

Dopo aver verificato che i due processi di *parsing* producevano lo stesso output relativo al dizionario informatizzato della classificazione Ateco 2007 a 5 digit, si è proceduto a controllare i risultati della fase di standardizzazione anche sul dizionario completo dell'Ateco.

Verificata la bontà dei risultati anche per questo caso, si è passati al caricamento delle basi informative relative ai due specifici dizionari della classificazione, utilizzando gli stessi file di *parsing* e la stessa strategia di codifica.

Infine è stata "importata" in CIRCE una caratteristica molto importante presente in ACTR v3, rappresentata dalla possibilità di personalizzare la strategia di codifica. In questo modo, i moduli afferenti ai diversi passi di standardizzazione, possono essere "combinati", in numero e sequenze, dall'utente, che può così adattare la strategia di codifica al particolare contesto applicativo, migliorando l'efficacia e la qualità dei risultati.

Un esempio di personalizzazione della strategia di codifica è l'uso ripetuto di un determinato passo di standardizzazione. In riferimento al contesto di codifica dell'Ateco 2007, il processo delle "*Double words" (Dwrd),* relativo al trattamento di coppie di parole, viene sottomesso per ben tre volte. Questa scelta venne effettuata per consentire la trasformazione corretta di descrizioni che contenevano al loro interno clausole di esclusione (molto frequenti in questa classificazione). Per queste stringhe, che non potevano essere abolite con il processo apposito (*deletion clauses*), ma che

dovevano comunque essere trattate, è stato possibile, grazie alla personalizzazione della strategia di codifica, trovare la trasformazione idonea ad evitare che generassero dei *match* non corretti. Un esempio è rappresentato dall'attività economica "*estrazione di minerali metallici non ferrosi, (escluso i minerali di uranio e di torio*)" per la quale era necessario, pur mantenendo la clausola di esclusione, far sì che le singole parole "*minerali*", "*uranio*" e "*torio*" non generassero *match* con altre attività che includevano l'estrazione di minerali non ferrosi.

E' importante sottolineare, che è stata proprio la flessibilità del software ACTR v3 a far si che non venisse sostituito in Istat con i prodotti che, pur rappresentando una sua evoluzione, non permettevano la personalizzazione della strategia di codifica. Questa caratteristica è adesso anche una peculiarità di CIRCE.

Al termine dei due processi appena descritti –preparazione ambiente di codifica a fase di standardizzazione - sono stati, quindi, (ri)creati in CIRCE due contesti :

1. il dizionario Ateco che prevede al suo interno solo Categorie associate a codici completi a 5 digit e che viene utilizzato per l'applicazione di codifica web;
2. il dizionario completo dell'Ateco che contiene anche codici con un dettaglio inferiore a 5 digit e che viene usato per la codifica automatica (*batch*) della variabile Ateco rilevata nelle indagini che non richiedono necessariamente una codifica al massimo dettaglio.

3.3 La fase di codifica

Nella fase di codifica vera e propria, come già spiegato nel paragrafo 2.5, la risposta testuale viene confrontata con il dizionario alla ricerca di un abbinamento esatto: se la ricerca fallisce, viene ricercato un abbinamento parziale ed in questo caso CIRCE individua, tramite un misura della similarità tra testi (S) "il codice" o "i codici" del dizionario con descrizione più simile alla risposta fornita dal rispondente. La misura di similarità è normalizzata ed assume valori compresi nell'intervallo [0,10] dove, se S=0 non si è verificato alcun abbinamento testuale mentre se S=10 si è verificato un abbinamento esatto. Gli *n* testi del dizionario con almeno una parola in comune con la risposta vengono poi ordinati per S decrescente.

La regione di accettazione per la misura di similarità si determina fissando tre parametri soglia: Smin, Smax, ΔS. Definire una strategia di codifica vuol dire proprio determinare i valori di questi parametri: scegliere parametri soglia molto elevati significherebbe prediligere l'accuratezza a scapito dell'efficacia, mentre definire parametri soglia troppo bassi inficerebbe la qualità della codifica. Per aiutare l'utente nella determinazione dei valori dei parametri il sistema prevede al suo interno una strategia di codifica, chiamata standard, che offre il giusto bilanciamento tra efficacia ed accuratezza, rispetto alla quale è ovviamente possibile cambiare i valori di uno o tutti i parametri in base alla esigenze di codifica e al tipo di classificazione.

La strategia utilizzata finora in Istat per le applicazioni di codifica *batch* è proprio quella standard che è basata sui seguenti valori dei parametri: Smax=8; Smin= 6; ΔS=0.2. In questo modo si massimizza l'efficacia senza penalizzare l'accuratezza.

Per l'applicazione di codifica web è utilizzata un'altra strategia che mantiene Smax=8 e Smin= 6, ma fissa un diverso valore della distanza: ΔS=1.5. Questa scelta è stata dettata dal fatto di cercare di proporre agli utenti web un ventaglio ampio di casistiche qualora l'attività dichiarata non produca un *match* diretto (unico).

Ciò premesso, sono stati effettuati, con i due software, dei primi passaggi di codifica *batch*, basati sulle due strategia di codifica, per ciascuno dei quali sono stati confrontati tempi, numerosità dei file di output, tassi di codifica e punteggi dei *match* prodotti con CIRCE rispetto ad ACTR v3. Il contesto utilizzato in entrambe le procedure è basato sul dizionario che contiene solo codici Ateco completi a 5 digit.

Per il primo confronto è stato utilizzato come file di input un insieme di testi di attività economica provenienti dal Censimento dell'Industria 2002 e costituito da 253 record; come strategia di codifica è stata usata quella standard (Smax=8; Smin= 6; ΔS=0,2). Le tabelle seguenti riportano l'esito delle procedure di codifica realizzate con CIRCE e con ACTR v3.

Tavola 4 – Codifica con ACTR v3

Risultato	Numero record	Percentuale
Unici	65	**25,69%**
Multipli	14	5,53%
Possibili	153	60,47%
Falliti	21	**8,30%**
Totale	253	100,00%

Tempo di codifica: 11,140 secondi
Match diretti: 12

Tavola 5 – Codifica con CIRCE

Risultato	Numero record	Percentuale
Unici	65	**25,69%**
Multipli	14	5,53%
Possibili	153	60,47%
Falliti	21	**8,30%**
Totale	253	100,00%

Tempo di codifica: 54,000 secondi
Match diretti: 12

Dalle tabelle si può osservare che i file di output (Unici, Multipli, Possibili e Falliti) risultano identici come numerosità; inoltre, in entrambe le procedure è stato prodotto lo stesso numero di *match* diretti (*match* unici con punteggio uguale a 10). Da un'analisi più dettagliata, relativa ai singoli file di output e finalizzata a verificare l'uguaglianza dei codici assegnati ed i relativi punteggi, è emerso quanto segue:

- **Confronto file Unici**: ai *match* unici e diretti è stato assegnato lo stesso codice da entrambe le procedure. Ai restanti unici è stato assegnato lo stesso codice con punteggio del *match* lievemente diverso a livello di decimali solo per alcuni casi. Ad esempio al testo di input "ATT.ORGANI LEGISL. ED ESECUTIVI, AMMINISTR. FINANZIARIA" sia CIRCE che ACTR v3 assegnano il codice corretto 84.11.1 relativo all'attività economica ATTIVITA' DEGLI ORGANI LEGISLATIVI ED ESECUTIVI, AMMINISTRAZIONE FINANZIARIA, ma mentre per CIRCE il punteggio del *match* è pari a 8,70 per ACTR v3 è 8.66. Questa lieve differenza è determinata dall'arrotondamento della fase di calcolo dei pesi. Data la non disponibilità dei sorgenti di ACTR v3 non è stato possibile conoscere la procedura di arrotondamento utilizzata;
- **Confronto file Multipli**: le due procedure hanno prodotto lo stesso numero di *match* Multipli (14 record) ed hanno anche generato lo stesso numero di record multipli (32 record) afferenti agli stessi codici;
- **Confronto file Possibili:** le due procedure hanno prodotto lo stesso numero di *match* Possibili (153 record) ma in CIRCE viene generato un record possibile in più (618 contro i 617 di ACTR v3) imputabile anche in questo caso all'arrotondamento della fase di calcolo dei pesi;
- **Confronto file Falliti:** le due procedure hanno prodotto lo stesso numero di *match* falliti associati alle stesse descrizioni testuali.

Il secondo passaggio di codifica è stato effettuato su un file di input più numeroso costituito dalle interrogazioni fatte dagli utenti dell'applicazione web (*query utente*) in data 02/01/2012. Il file, costituito da 7.741 record, è stato sottomesso ai due sistemi di codifica utilizzando questa volta i

parametri propri dell'applicazione web (Smax=8; Smin= 6; ΔS=1,5).

Tavola 6 – Codifica con ACTR v3

Risultato	Numero record	Percentuale
Unici	3.831	**49,49%**
Multipli	493	6,36%
Possibili	2.708	34,98%
Falliti	709	**9,17%**
Totale	7.741	100,00%

Tempo di codifica: 24,250 secondi
Match diretti: 3054

Tavola 7 – Codifica con CIRCE

Risultato	Numero record	Percentuale
Unici	3.829	**49,46%**
Multipli	494	6,38%
Possibili	2.708	34,98%
Falliti	710	**9,17%**
Totale	7.741	100,00%

Tempo di codifica: 600 secondi
Match diretti: 3053

Dai risultati è possibile notare che i tempi di codifica di CIRCE sono accettabili (anche aumentando il numero di record del file di input) e che le numerosità dei file di output risultano abbastanza simili. Scendendo più in dettaglio:

- **Confronto file Unici:** CIRCE ha prodotto un *match* diretto in meno rispetto ad ACTR v3. Dopo un'ulteriore fase di addestramento di CIRCE l'errore è stato risolto e adesso viene prodotto un *match* diretto. L'analisi sui *match* unici con punteggio inferiore a 10 ha evidenziato solo una leggera differenza nei punteggi dei *match*;
- **Confronto file Multipli:** il record multiplo in più prodotto da CIRCE è stato risolto con una piccola modifica fatta all'interno di una funzione del pacchetto CIRCE;
- **Confronto file Possibili:** non ci sono casi da analizzare;
- **Confronto file Falliti:** l'unico record da analizzare, in quanto fallito per CIRCE ed unico per ACTR v3, è stato risolto con la fase di addestramento citata nel confronto tra gli Unici.

I risultati dei primi test effettuati mostrano la bontà del lavoro fatto sia relativamente all'implementazione dell'algoritmo di *matching*, che ricalca perfettamente quello di ACTR v3, sia alla migrazione e al controllo di tutto l'ambiente di codifica.

3.4 I test sulla qualità e sull'efficacia

Il software ACTR v3 ha rappresentato il fulcro del sistema web di interrogazione, ricerca e codifica dell'ATECO 2007 pubblicato sul sito dell'Istituto che è diventato, nel corso degli anni, uno strumento irrinunciabile per le procedure amministrative che necessitano del codice di attività economica.

La testimonianza dell'ampio utilizzo del sistema è il milione, circa, di interrogazioni web effettuate dagli utenti del sito durante gli anni 2013 e 2014 così come mostra la tavola 8.

Tavola 8 - *Query* digitate dagli utenti web per mese negli anni 2013-2014

Mesi	Anno 2013	Anno 2014
Gennaio	77.289	75.647
Febbraio	76.358	84.381
Marzo	76.323	108.648
Aprile	85.671	70.867
Maggio	61.675	70.377
Giugno	65.628	84.105
Luglio	81.240	66.797
Agosto	33.580	48.461
Settembre	70.454	73.485
Ottobre	66.269	76.342
Novembre	62.214	65.264
Dicembre	68.770	90.751
Totale	**825.471**	**915.125**
Media settimanale	***15.874***	***17.599***

La costruzione dell'ambiente di codifica automatica dell'Ateco con il software ACTR v3 è iniziata in Istat nel 1998 con una collaborazione tra gli esperti della classificazione e gli esperti del sistema di codifica. La Classificazione ufficiale dell'Ateco di riferimento era quella del 1991. Nel corso degli anni il sistema di codifica automatica è stato aggiornato alla Classificazione ufficiale Ateco 2002 fino all'attuale del 2007. Quest'ultimo contesto è stato utilizzato per la codifica *batch* dei testi relativi all'attività economica rilevati nei Censimenti[8] ed in varie altre indagini dell'Istituto, con buoni risultati sia a livello di tassi di codifica che di qualità della stessa. Verso la metà del 2008 il contesto di codifica creato con ACTR v3 è stato disponibile sul sito web dell'Istat fino a giugno 2014 quando si è resa necessaria la sostituzione di ACTR v3 con il nuovo pacchetto di codifica CIRCE.

La sostituzione del software ha comportato la partecipazione di diverse figure professionali impegnate nelle varie fasi di lavoro. Il nuovo pacchetto di codifica, infatti, doveva, garantire le stesse prestazioni di ACTR v3 sia in termini di efficacia che di accuratezza, ossia in termini degli indicatori di qualità usati solitamente per la valutazione dei sistemi di codifica automatica:

- l'efficacia (*recall rate*), che rappresenta la percentuale dei testi codificati univocamente sul totale dei testi da codificare;
- l'accuratezza (*precision rate*), che indica la percentuale dei codici corretti sul totale dei codici univoci assegnati ai testi.

Il compito di creare un software che potesse uguagliare ACTR v3 è stato molto arduo in quanto l'efficacia e l'accuratezza di ACTR v3 sono risultate sempre molto alte e sono anche migliorate nel tempo, grazie al lavoro di monitoraggio, addestramento ed aggiornamento dell'ambiente di codifica svolto fino alla metà del 2014.

Alla realizzazione del nuovo pacchetto di codifica è seguita una fase di test sulla qualità di CIRCE prima del suo rilascio.

I test ed i relativi risultati sono descritti nelle pagine successive.

3.4.1 L'efficacia e l'accuratezza di ACTR v3: un benchmark per CIRCE

Le prestazioni di ACTR v3, in termini di efficacia ed accuratezza, costituivano il traguardo da

[8] Censimenti popolazione: indagini pilota del 1991 e del 2001 e convivenze 2001. Censimento dell'industria e servizi: intermedio del 1998 e 2001

raggiungere con CIRCE.

Come parametri di riferimento sono stati utilizzati quelli derivanti dell'ultimo monitoraggio disponibile per ACTR v3 (Allegato 1) e basato sulle stringhe di attività economica digitate dagli utenti web dal 2008 al 2013 (circa 4,5 milioni di record).

Prima di realizzare la codifica con ACTR v3, l'insieme di stringhe è stato sottoposto (così come previsto nella procedura di monitoraggio) ad un passaggio di *parsing* semplificato per rimuovere gli elementi grammaticali che possono rendere diverse due descrizioni di fatto uguali. Successivamente i dati sono stati elaborati con un programma SAS al fine di produrre un data set contenente solo le descrizioni diverse (707.339) con accanto la relativa frequenza. Il data set risultante è riportato nella tabella seguente, dove, per ogni classe di frequenza, sono indicati il numero di descrizioni diverse ed il numero di quelle originali.

Tavola 9 - Testi di input per classi di frequenza

Classe di frequenza testo	Descrizioni diverse	Descrizioni originali
1	464.107	464.107
2-8	202.467	623.261
9-30	26.936	412.189
31-50	4.935	192.220
51-100	4.010	280.940
101-300	3.116	523.507
301-500	730	279.683
501-1.000	574	396.646
1.001-2.000	250	350.684
2.001+	214	1.043.766
Totale	**707.339**	**4.567.003**

Per individuare i valori di *benchmarck* di efficacia e di accuratezza di CIRCE, è stato realizzato, su questo insieme di stringhe, un passaggio di codifica con ACTR v3, ottenendo i risultati contenuti nelle tabelle seguenti (Tavole 10 e 11):

- la tavola 10 mostra che l'efficacia di ACTR v3, ossia il *recall rate*, è pari al 55,1% (Unici) e che in generale il sistema riesce ad individuare un codice unico corrispondente alla descrizione digitata oppure individua un ventaglio di possibilità (Multipli e Possibili) tra cui l'utente può trovare il codice cercato nell' 89,8% dei casi;
- la successiva tavola 11 mostra invece il livello di precisione rappresentato, per questo progetto, dalla quota di *match* che all'interno degli Unici hanno ricevuto un punteggio pari a 10 (*match* diretti): il *precision rate* si attesta ad un livello alto e pari al 75%.

Tavola 10 - Efficacia (*recall rate*) di ACTR v3 giugno 2014

	Numero di descrizioni originali	
	Valori assoluti	Valori %
Unici	2.517.600	**55,1%**
- di cui con codice Ateco n.c. e/o <5 digit*	*561.483*	*12,3%*
Multipli	117.842	2,6%
Multipli con 5 codici Ateco uguali	4.765	0,1%
Possibili	1.433.236	31,4%
Possibili con 5 codici Ateco uguali	26.408	0,6%
Falliti	467.152	10,2%
Totale	**4.567.003**	**100,00%**

*****n.c: codice fittizio associato ai testi non codificabili in quanto troppo generici o non riconducibili ad un'attività economica

Tabella 11 – Accuratezza (*precision rate*) di ACTR v3 giugno 2014

	Punteggio del match	Numero di descrizioni originali	
		Valori assoluti	Valori %
Unici	8 e 9	628.851	25,0%
	10	1.888.749	**75,0%**
Totale		**2.517.600**	**100,00%**

3.4.2 L'efficacia di CIRCE

Per verificare che CIRCE garantisse almeno lo stesso *recall rate* di ACTR v3 occorreva realizzare la codifica sul data set precedentemente descritto.

Poiché, come già detto nel capitolo 2, CIRCE non garantisce la stessa velocità di esecuzione propria dei pacchetti scritti con linguaggi compilati, è stato necessario utilizzare un data set più piccolo contenente solo i record relativi a descrizioni di attività economica con frequenza maggiore di 2. Il data set utilizzato per il confronto è composto da 135.271 descrizioni diverse tra loro corrispondenti a 3.886.974 descrizioni originali.

Questo insieme di testi è stato sottoposto alla codifica *batch* con entrambi i prodotti, utilizzando come contesto di codifica quello aggiornato al giugno 2014. I risultati sono riportati nella tavola 12.

Tavola 12 - Efficacia (*recall rate*): CIRCE verso ACTR v3

Output della codifica automatica	Codifica con CIRCE			Codifica con ACTR v3		
	Numero di descrizioni originali	Numero di descrizioni diverse tra loro	% su numero descrizioni originali	Numero di descrizioni originali	Numero di descrizioni diverse tra loro	% su numero descrizioni originali
Unici	2.355.039	48.764	**60,60%**	2.354.903	48.739	**60,60%**
- di cui Unici con codice Ateco n.c. e/o <5 digit	*528.740*	*5.793*	*13,60%*	*528.727*	*5.791*	*13,60%*
Multipli	111.139	2.910	2,90%	111.035	2.901	2,90%
Multipli con 5 codici Ateco uguali	4.729	34	0,10%	4.729	34	0,10%
Possibili	1.139.100	58.102	29,30%	1.139.262	58.120	29,30%
Possibili con 5 codici Ateco uguali	19.986	1.547	0,50%	20.000	1.550	0,50%
Falliti	256.981	23.914	6,60%	257.045	23.927	6,60%
Totale	**3.886.974**	**135.271**	**100,00%**	**3.886.974**	**135.271**	**100,00%**

Nonostante CIRCE abbia impiegato un tempo elevato per la codifica in *batch*[9], si può osservare che il tasso di codifica ottenuto è perfettamente identico, in termini percentuali, a quello di ACTR v3: entrambi i pacchetti codificano il 60,6% dei testi in modo univoco[10].

Osservando i valori assoluti si riscontrano, invece, lievi differenze. Si è proceduto pertanto ad un'ulteriore fase di analisi per verificare quanti codici diversi fossero stati assegnati dai due sistemi alla stessa descrizione. I casi analizzati hanno riguardato le descrizioni testuali con "destinazione" diversa tra le due applicazioni e precisamente:

- codici Unici assegnati da entrambe le applicazioni, ma con codice diverso (nessun caso presente);
- codici Unici assegnati da ACTR v3 e codici "Non Unici" assegnati da CIRCE e viceversa;
- codici Falliti assegnati da ACTR v3 e codici "Non Falliti" assegnati da CIRCE e viceversa.

[9] I tempi di codifica *batch* di CIRCE potranno essere ridotti in seguito alla riscrittura di alcune porzioni del codice sorgente finalizzata a elaborare i processi in parallelo sfruttando il numero di CPU presenti sui pc. Questo dovrebbe permette di superare le problematiche legate alla codifica *batch* ed anche alla fase di monitoraggio del sistema per la quale, comunque, data la mole di dati elaborati, potrebbe rendersi necessaria una riprogettazione architetturale non potendo CIRCE garantire, per sua natura, gli stessi tempi di esecuzione di ACTR v3.

[10] E' importante spiegare che la percentuale di Unici di ACTR v3 è più alta di quella ottenuta sul data set completo (4,5 milioni di record circa), in quanto il sistema è stato addestrato principalmente per la codifica delle descrizioni di attività economica con frequenze elevate. Questo sia per mancanza di risorse sia, e soprattutto, per ottimizzare in tempi brevi le risposte del servizio web alle richieste di codifica più frequentemente effettuate di dagli utenti del sito.

Dall'analisi è emersa una differenza minima, pari a circa lo 0,021% sull'universo codificato, che in termini assoluti corrisponde a 122 record (ossia ad 818 descrizioni originali). Tale risultato è da imputare prevalentemente alle procedure di arrotondamento nel calcolo del punteggio usate dai due sistemi, che, per piccolissime differenze a livello di decimali, modificano i limiti delle regioni di accettazione dei punteggi spostando i codici Unici nella regione dei codici Multipli e viceversa.

3.4.3 L'accuratezza di CIRCE

Per valutare l'accuratezza (*precision rate*) di CIRCE è stato utilizzato il nuovo ambiente di codifica che si è venuto a creare durante le fasi di sviluppo, nel corso delle quali venivano effettuati continui confronti con ACTR v3. Le differenze tra i due pacchetti venivano sanate di volta in volta apportando anche correzioni ai file di *parsing* o al dizionario perché relativi, ad esempio, a sinonimi o empiriche non corretti in generale che costituivano la causa delle differenze di codifica fra i due sistemi.

Il nuovo ambiente così aggiornato sarebbe stato alla base dell'applicazione di codifica web.

La valutazione dell'accuratezza è stata effettuata sulla base di un campione casuale di testi estratto dall'universo completo di 4,5 milioni di *query* degli utenti web. L'estrazione del campione è avvenuta dopo aver stratificato l'universo per classi di frequenza, secondo una procedura di campionamento già seguita per la valutazione della qualità della codifica con ACTR v3.

La dimensione campionaria totale è stata determinata trattando i singoli strati come indipendenti ed individuando per ciascuno la dimensione ottimale rispetto alla percentuale attesa di codici unici all'interno di ciascuno strato. Come percentuali attese sono state prese quelle di ACTR v3 per la codifica dei 4.567.003 di *query* utente (Tavola 13).

Tavola 13 – ACTR v3: Percentuali di unici per classe di frequenza

Classe di frequenza testo	Unici	Totale testi	Percentuali di Unici per classe (P_h)
1	105.907	464.107	23,00%
2-8	186.645	623.261	30,00%
9-30	182.811	412.189	44,00%
31-50	101.838	192.220	53,00%
51-100	158.642	280.940	56,00%
101-300	316.241	523.507	60,00%
301-500	168.260	279.683	60,00%
501-1.000	261.062	396.646	66,00%
1.001-2.000	245.853	350.684	70,00%
2.001+	790.341	1.043.766	76,00%
Totale	**2.517.600**	**4.567.003**	

In particolare, la numerosità di ogni strato, ossia n_h, è stata determinata con la seguente formula:

$$n_h = n_{0h}/1 + (n_{0h} - 1)/N_h \qquad (4)$$

dove

$$n_{0h} = z^2_{1-\alpha/2}\tilde{P}_h(1-\tilde{P}_h)/d_h^2 \qquad \text{con h=1,…M}$$

Nelle (4):

- $\tilde{P}_h$ rappresenta la percentuale di unici attesa all'interno dello strato h;
- d_h rappresenta il margine di errore consentito per la stima di P_h;
- z è il percentile della distribuzione normale standardizzata e tale che $Pr(\tilde{P}_h\text{-}P_h >= d_h) = \alpha$.

Sommando le numerosità ($n_1+n_2+\ldots n_M$) dei singoli strati determinate con la (4), si è ottenuto un campione di numerosità totale pari a 5.168 record, come indicato nella tavola seguente.

Tavola 14 - Campione stratificato per classe di frequenza *query* utente

Classi di frequenza	Testi originali	Testi originali e diversi	Precisione codifica richiesta per classe	Errore	Campione	Campione (correzione ultimi strati)
		N_h	P_h	d_h	n_h	n_h *(corretto)*
1	464.107	464.107	0,23	0,04	422	422
3-8	623.261	202.467	0,29	0,04	488	488
9-30	412.189	26.936	0,43	0,045	456	456
31-50	192.220	4.935	0,53	0,045	431	431
51-100	280.940	4.010	0,56	0,03	833	833
101-300	523.507	3.116	0,60	0,03	770	770
301-500	279.683	730	0,60	0,01	677	730
501-1.000	396.646	574	0,66	0,01	538	574
1.001-2.000	350.684	250	0,69	0,01	243	250
2.001+	1.043.766	214	0,71	0,01	208	214
Totale	**4.567.003**	**707.339**			**5.068**	**5.168**

La codifica *batch* del testi del campione ha restituito i seguenti risultati:

Tavola 15 - Risultati codifica con CIRCE sul campione di testi

Risultato	Numero record	Percentuale
Unici	2.786	**53,90%**
Multipli	146	2,80%
Possibili	1.726	33,40%
Falliti	510	9,90%
Totale	**5.168**	**100,00%**

Per la valutazione della precisione della codifica, in modo analogo a quanto fatto per ACTR v3 (Tavola 11), occorre suddividere i codici unici in funzione del punteggio di *match*. Nel campione analizzato, sono risultati 2.786 codici unici, di cui 1.907 hanno prodotto un *match* diretto (punteggio 10) ed i restanti 879 un *match* indiretto.

Tabella 16 – Accuratezza (*precision rate*) di CIRCE

	Punteggio del match	Numero di record
Unici	8 e 9	879
	10	1.907
Totale		**2.786**

Per calcolare il livello di accuratezza, come la quota dei *match* unici diretti (punteggio 10) sul totale dei codici unici, occorre riportare i dati all'universo di riferimento. La tabella che segue mostra la distribuzione dei *match* diretti per classe di frequenza e la consistenza delle classi in termini di codici unici.

Tavola 17 - Accuratezza CIRCE: distribuzioni degli match diretti per classe di frequenza

Classe di frequenza testo	Match diretti	Totale Record Unici	Proporzione di mach diretti per classe	Frequenze unici (pesi)
1	17	109	0,16	109
2-8	52	153	0,34	491
9-30	111	201	0,55	3.226
31-50	136	229	0,59	8.942
51-100	352	500	0,70	34.814
101-300	316	447	0,71	74.574
301-500	332	439	0,76	169.793
501-1.000	306	383	0,80	262.026
1.001-2.000	142	174	0,82	247.618
2.001+	143	151	0,95	787.806
Totale	**1.907**	**2.786**		**1.589.399**

Dopo aver calcolato la media ponderata delle proporzioni di *match* diretti per classi di frequenza, con i pesi dati dalle frequenze di ciascuna classe, si ottiene un livello di accuratezza del sistema CIRCE pari al'86,5%.

Quindi è possibile affermare che anche per questo aspetto CIRCE risulta essere totalmente in linea con ACTR v3, che ha registrato una quota di *match* diretti pari al 75% (Tavola 11), e che, grazie all'aggiornamento del contesto di codifica, viene offerto agli utenti un livello maggiore di qualità del servizio.

Per una determinazione più ampia del livello di qualità di CIRCE è stata valutata anche la correttezza dei *match* indiretti (dato non disponibile per ACTR v3), ossia dei codici unici con un punteggio di *match* inferiore a 10[11].

Analizzando la qualità di questi ultimi emerge che, nella maggioranza dei casi (672), i codici sono stati assegnati correttamente, così come mostra la tabella che segue.

Tavola 18 - CIRCE: Analisi di qualità match indiretti

	Numero record
Totale Unici	2.786
Match diretti (unici con punteggio pari a 10)	**1.907**
Match indiretti (unici con punteggio minore di 10)	879
Di cui:	
• con **codifica corretta**	**672**
• con codifica errata	63
• testi non codificabili (testo generico)	144

Anche in questo caso, per esprimere il livello di accuratezza allargato ai *match* indiretti e corretti è stato necessario riportare i dati all'universo di riferimento, in modo analogo a quanto fatto per i *match* diretti.

[11] Prassi generalmente seguita per la valutazione della qualità delle applicazioni di codifica *batch* con ACTR v3.

Tavola 19 - Accuratezza CIRCE: distribuzioni degli unici corretti per classe di frequenza

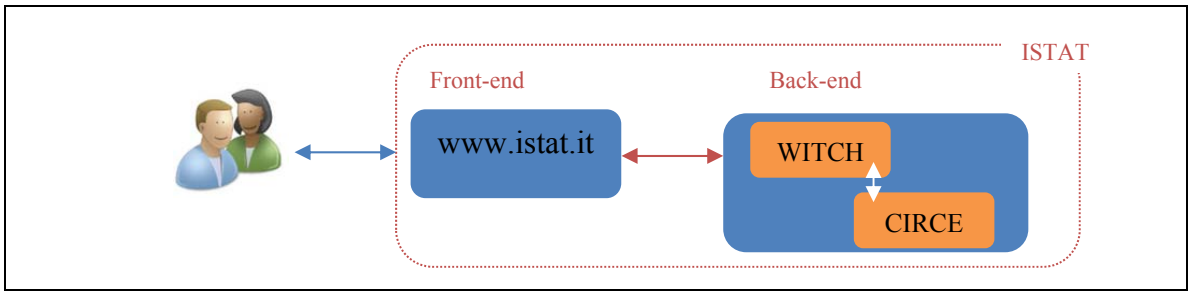
Classe di frequenza testo	*Match* diretti	*Match* indiretti e corretti	Totale unici corretti	Totale Record Unici	Proporzione di codici corretti per classe	Frequenze unici (pesi)
1	17	82	99	109	0,91	109
2-8	52	87	139	153	0,91	491
9-30	111	74	185	201	0,92	3.226
31-50	136	76	212	229	0,93	8.942
51-100	352	112	464	500	0,93	34.814
101-300	316	100	416	447	0,93	74.574
301-500	332	74	406	439	0,92	169.793
501-1.000	306	45	351	383	0,92	262.026
1.001-2.000	142	19	161	174	0,93	247.618
2.001+	143	3	146	151	0,97	787.806
Totale	**1.907**	**672**	**2579**	**2.786**		**1.589.399**

Dopo aver calcolato la media ponderata delle proporzioni di unici corretti per classi di frequenza si ottiene un livello di accuratezza del sistema CIRCE pari al 94,7% che conferma la bontà del nuovo sistema di codifica.

4. La nuova applicazione web

L'applicazione di codifica on-line è un servizio web utilizzato dalla pagina "Classificazione delle attività economiche Ateco 2007" del sito istituzionale[12]. L'utente dopo avere digitato la stringa nella apposita sezione di ricerca , effettua una richiesta ad un altro sistema web che risiede su un server Istat di back-end. Tale sistema consta di due applicativi: WITCH (Web service Interface To Coding Handler), applicazione web in php e CIRCE (Comprehensive Istat R Coding Environment), applicativo di codifica in R. I due sistemi, completamente integrati, mettono a disposizione dell'utente uno strumento efficiente di codifica, in termini di tempi di risposta e di qualità dei risultati.

Figura 10 - Sistema di codifica ATECO 2007



La nuova applicazione web di back-end è stata completamente riprogettata e realizzata con una differente architettura rispetto alla precedente basata su ACTR v3, allo scopo di migliorare le qualità applicative: le funzionalità, la sicurezza, l'affidabilità, la portabilità e la riusabilità. Il sistema fornisce, infatti, un servizio di codifica (**web service**) che potrà essere riutilizzato in altri contesti applicativi.

[12] http://www.istat.it/it/strumenti/definizioni-e-classificazioni/ateco-2007

4.1. Architettura e struttura sofware

L'applicazione è costituita da una *web application* realizzata con una architettura distribuita. Lo scopo è quello di fornire all'utente (interno od esterno) un'interfaccia *web based* simile a quella precedente con funzionalità migliorate ed un layout rinnovato, tramite cui interrogare il motore di codifica per l'Ateco (CIRCE).

Sebbene l'interfaccia utente sia funzionalmente simile a quella adottata in passato, è implicito che l'architettura attuale presenti peculiarità appositamente introdotte, allo scopo di consentire al nuovo motore di codifica CIRCE di ricevere input utente e fornire output di codifica. E' quindi essenziale, per la comprensione dell'architettura attuale, sapere che l'applicazione prevede una serie di soluzioni studiate sia per offrire continuità di servizio agli utenti sia per risolvere problematiche inerenti la sicurezza che un sistema di questo tipo deve garantire in esercizio. Si esamini a questo scopo la figura 11 che contiene lo schema architetturale.

Nel dettaglio si individuano i seguenti blocchi e componenti dell'applicazione:

1. Front-end server
2. Back-end/Application server
3. DBMS server

Il Front-end server è composto da hardware dedicato su cui risiede il sito istituzionale Istat. Il sito medesimo è realizzato attraverso una distinta *web application* che sfrutta un framework e un ambiente di esecuzione con engine di scripting e web server presenti sul front-end server. Nella pagina "Classificazione delle attività economiche Ateco 2007" è stata creata una pagina con un form per l'immissione di stringhe da codificare attraverso CIRCE (paragrafo 4.2.1).

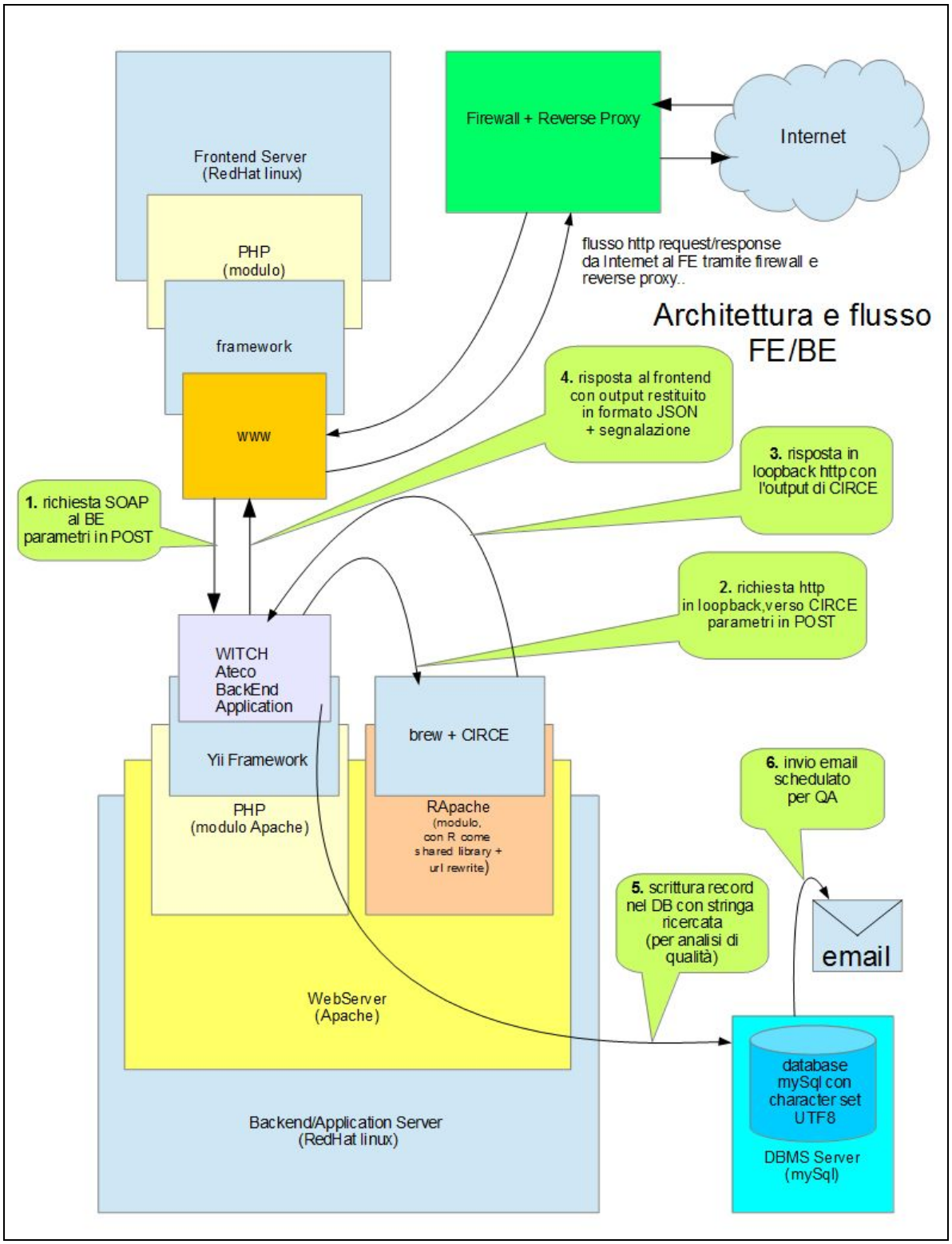
Il Front-end server contiene un client per collegarsi all'applicazione di back-end ed inoltrare la richiesta.

Il Back-end/Application server costituisce il nucleo portante dell'applicazione, in quanto ospita sia una *web application* di back-end, realizzata in linguaggio php tramite il framework Yii denominata WITCH, sia l'ambiente di esecuzione R ed il motore di codifica CIRCE, pacchetto realizzato in linguaggio R.

È da notare che, essendo i componenti ed il software Yii e R logicamente distinti ed essendovi interazioni tra essi basate su tecnologia web, tali ambiti potrebbero facilmente essere ospitati su server diversi senza apportare modifiche all'applicazione. In fase di realizzazione del sistema si è operata la scelta di accentrare entrambi su un unico server principalmente per ragioni di razionalizzazione delle risorse e per evitare la latenza addizionale che sarebbe conseguita alla comunicazione in rete tra macchine distinte.

Il DBMS server è costituito dal server DBMS standard di produzione MySql; esso viene utilizzato per ospitare un DB che contiene dati sulle ricerche effettuate dagli utenti (*query* utente). Ciò allo scopo di recuperare periodicamente (su base settimanale) le stringhe immesse dagli utenti ed inviarle ai referenti di CIRCE e della classificazione Ateco per essere sottoposte all'analisi di qualità finalizzata a migliorare nel tempo l'efficacia del sistema di codifica.

Figura 11 - Architettura software

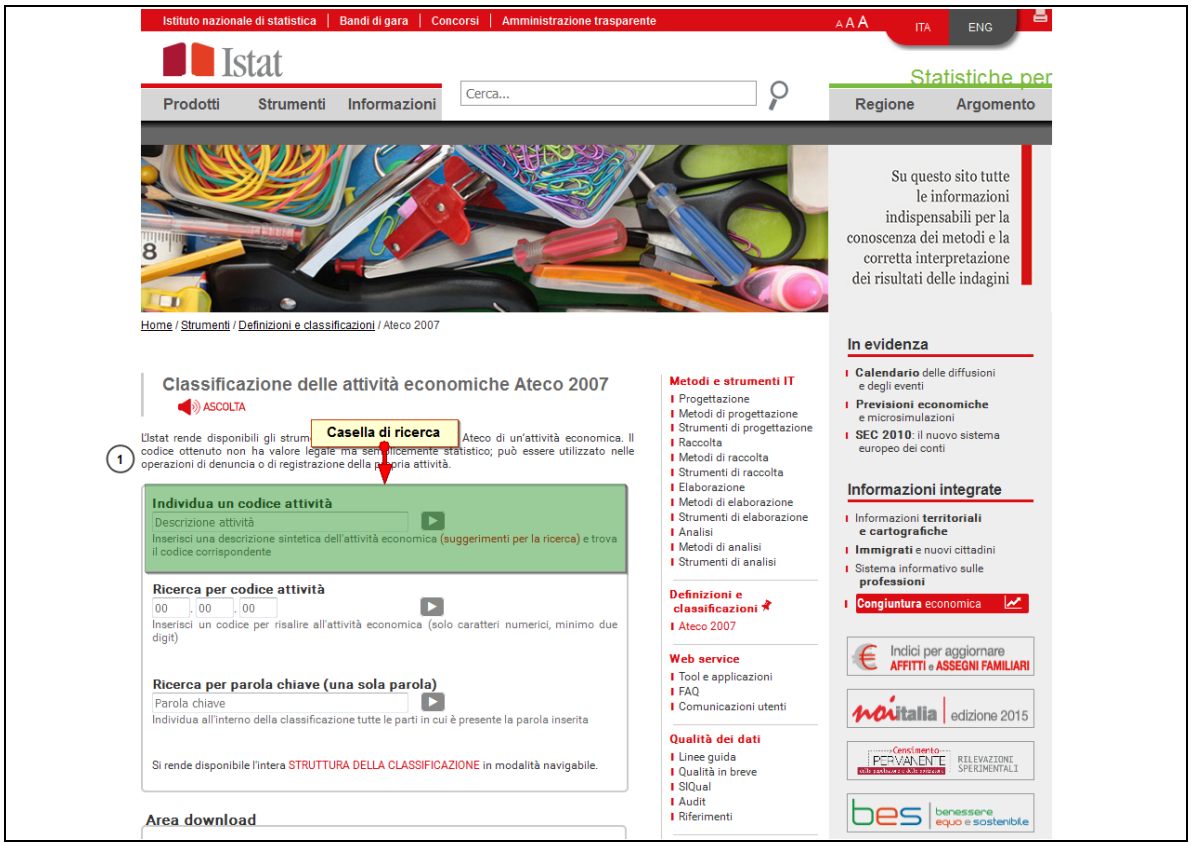


4.2. Funzionamento e flusso dati

4.2.1. Richiesta utente

Quando l'utente intende effettuare una o più richieste di codifica al sistema, accede ad una pagina specifica sul sito istituzionale (figura 12).

Figura 12 - Front-end



La descrizione "Individua un codice di attività" indica il campo di testo da utilizzare per l'immissione dell'input: l'utente inserisce la stringa da codificare relativa alla attività economica di interesse, ad esempio "produzione di gelato", ed effettua un click sul pulsante di ricerca.

Questa operazione genera una chiamata con dati in POST tramite un proprio client SOAP verso l'applicazione WITCH (punto 1 - figura 11) che provvede a gestire le fasi successive.

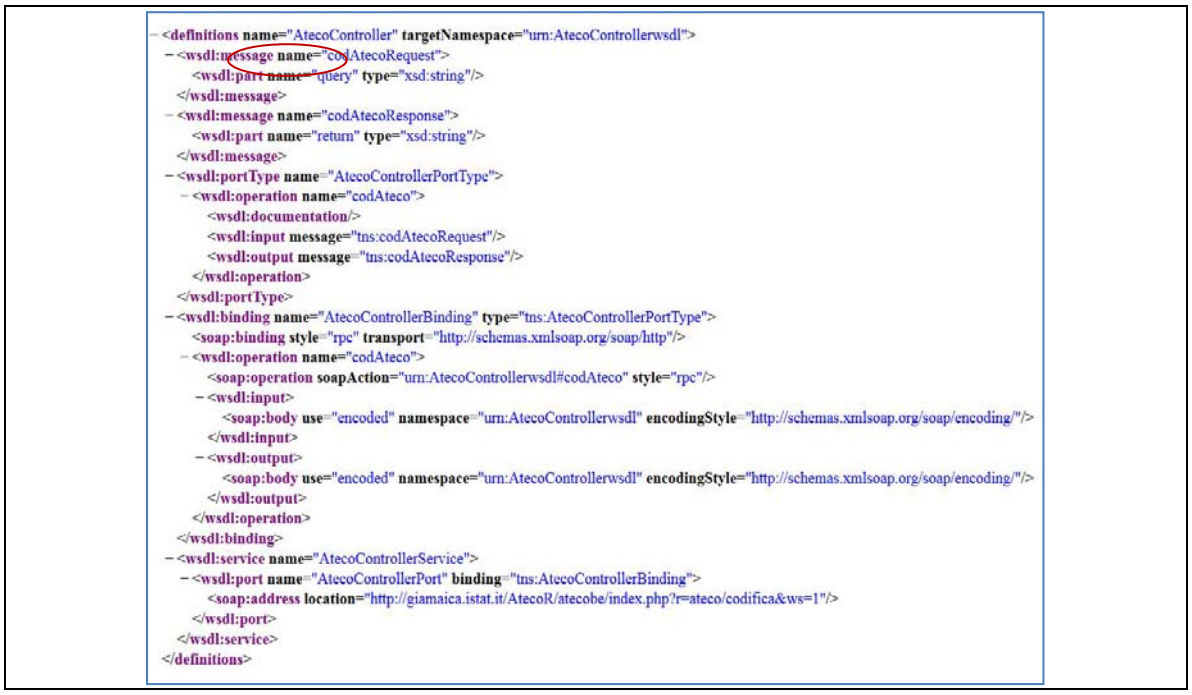
4.2.2. Richiesta a WITCH

La richiesta del punto 1 arriva all'Apache web server della macchina di back-end ed in particolare all'applicazione WITCH. E' stato implementato un filtro applicativo forte volto a prevenire un utilizzo arbitrario del sistema di codifica. Infatti, come descritto in dettaglio di seguito, WITCH effettua una serie di manipolazioni sulla stringa di input (*security policy B*) in modo da sottoporre a CIRCE solo stringhe valide, ovvero depurate da elementi potenzialmente dannosi per il sistema senza alterare la significatività delle stringhe di input.

La validazione della stringa di input, con eventuale bonifica/escaping/controlli, viene effettuata per evitare attacchi mediante *injection* di comandi e codice (sia a livello php che R e XSS); successivamente l'applicazione avvia la fase di smistamento verso l'application server che ospita R ed il pacchetto CIRCE.

Nello specifico, l'applicazione WITCH è composta da software php/Yii che implementa un Web Service basato sul protocollo SOAP. Il file WSDL associato a WITCH è il seguente:

Figura 13 - file WSDL associato a WITCH



Dal file WSDL si evince che è possibile effettuare una RPC (Remote Procedure Call) verso una funzione invocata quale "ateco/codifica" che è mappata su un metodo della classe AtecoController presente in WITCH, metodo che effettua di fatto lo svolgimento delle operazioni connesse alla codifica di una stringa di input passata dal front-end.

Il metodo in questione si occupa dapprima di validare/bonificare la stringa. Per incrementare la robustezza del sistema verso tipologie di attacco XSS reflected e command/code injection, la stringa viene sottoposta ad una serie di manipolazioni, effettuate anche tramite espressioni regolari, di seguito indicate:

1. *tag removing*: i caratteri '<' di apertura tag vengono sostituiti dalla stringa 'INFERIORE' ed i caratteri '>' di chiusura tag vengono sostituiti dalla stringa 'OLTRE';
2. *whitelisting*: i caratteri che rientrano nel pattern /([^a-zA-ZàáèéìíòóùúÀÁÈÉÌÍÒÓÙÚ\s])/u vengono mantenuti (OWASP compliant);
3. *rimozione accenti*: i caratteri accentati vengono convertiti in equivalenti non accentati (per CIRCE che opera su caratteri non accentati);
4. *quoting*: le stringhe risultanti vengono quotate in doppi apici (");
5. *uppering*: le stringhe vengono convertite in maiuscolo (uppercase) .

Successivamente viene preparato un array di parametri da passare a CIRCE: pathbase, funzione, progetto, stringa. I primi tre sono prefissati secondo le indicazioni del team di sviluppo di CIRCE e variano solo in funzione dell'installazione del pacchetto di codifica; il parametro "stringa" invece è costituito dalla stringa di input alla quale sono stati applicati i passi da 1 a 5 precedentemente descritti, indicata come *clean_query*.

L'array è utilizzato per creare una richiesta (http request) verso il sottosistema che ospita CIRCE.

4.2.3. Richiesta a CIRCE

La richiesta http, che contiene i parametri di ricerca (punto 2 - figura 11), parte dall'applicazione WITCH verso l'application server (in loopback) puntando ad una pagina html speciale (brewing R). In pratica l'application server contiene il sottosistema che ospita CIRCE ed è composto da:

- Apache, web server;
- rApache, sistema per la scrittura di *web application*s in R che usa handlers R configura-

bili;

- Brew, templating engine ed handler per R;
- R, interprete e librerie.

rApache è un modulo di Apache che contiene tutto il necessario per creare in memoria un interprete R: tale metodologia sfrutta il meccanismo degli shared objects di linux, ovvero una libreria/modulo (mod_R.so) caricato in memoria nello spazio di indirizzamento di Apache e tale da consentire l'esecuzione di script R senza la necessità di creare processi separati, che necessiterebbero, ad esempio, della shell o script da eseguire tramite exec ed altre system call (*security policy C*) del sistema operativo. Questa soluzione, vantaggiosa anche in termini di efficienza, offre il vantaggio di non dover richiamare funzioni di sistema potenzialmente sfruttabili per scopi illeciti (scrittura su filesystem di contenuti arbitrari, creazione di processi, esecuzione di comandi ecc. invocando system o shellexec o exec o altro). Inoltre rApache, quale modulo di Apache, in un sistema stateless offre il vantaggio della gestione intrinseca del parallelismo in esecuzione offerta dal web server. In parole povere è Apache che si occupa delle richieste in arrivo: esse sono parallele e indipendenti e la gestione delle code è affidata al normale flusso di esecuzione nel web server e non è gestita da codice applicativo. Si noti, invece, che nella precedente applicazione web le richieste erano state serializzate tramite la creazione e rimozione di un lock esclusivo basati su file. L'utilizzo del parallelismo in esecuzione risulta vantaggioso in termini di utilizzo della macchina ed in previsione dei test di carico sul sistema e della futura scalabilità.

Brew è un template framework per rApache/R in Apache. Esso consente la scrittura di file 'speciali' .html che contengono scripting tags brew. Nell'attuale applicazione brew è utilizzato come entry point al sottosistema di codifica verso CIRCE. La configurazione di brew prevede che ad un path specifico interno alla document root del web server venga assegnato l'handler brew, in modo tale che i file .html ivi presenti siano interpretati da brew (brewing R).

Un ulteriore aspetto rilevante per la sicurezza applicativa (*security policy D*) è che non si effettuano scritture sul filesystem. Infatti, il testo da ricercare viene inviato come parametro in POST nella richiesta al web server lavorando così esclusivamente in memoria. Naturalmente, essendovi interpreti R coinvolti, il processo è per definizione CPU intensive e quindi la memoria utilizzata costituisce un parametro importante, ma secondario, rispetto al consumo di CPU.

L'assenza di scritture su filesystem è possibile in quanto il brewing R consente il passaggio di parametri allo script R caricato dinamicamente e la restituzione dei risultati "a video" (stdout). Questo meccanismo è analogo al re-indirizzamento dell'output standard di Unix ed è gestito internamente da rApache e brew.

4.2.4. Onion style execution

La http request inviata da front-end arriva a WITCH e viene propagata da WITCH (a meno della validazione/bonifica dell'input) verso CIRCE in loopback; viene quindi intercettata da Apache ed essendo la request relativa ad un path specifico associato all'handler brew questo viene invocato ed il template brew associato viene eseguito ricevendo i parametri (in POST) propagati da WITCH inclusa la stringa da codificare; tramite rApache l'interprete brew carica R come shared object, carica la libreria di codifica di CIRCE ed avvia l'esecuzione della funzione di codifica.

A questo punto l'interprete R in memoria esegue il codice relativo alla funzione di codifica ritornando, in caso di successo, una stringa contenente un array (formato JSON con codici ateco, descrizioni brevi etc.) con l'esito della ricerca. Di seguito il JSON ritornato nel caso della stringa di ricerca "produzione di gelato" :

{"search_string":""PRODUZIONE DI GELATO"","risultato":"unici","0":

[{"ateco_code":"10.52.0","ateco_description":"FABBRICAZIONE DI GELATI"}]}

Si riporta per completezza l'esito delle ricerche 'carpentiere' (multipli) e 'produzione' (troppo generica e che produce zero risultati) ed impiegato (falliti)

{"search_string":""CARPENTIERE"","risultato":"possibili","0":[{"ateco_code":"25.11.0","ateco_description":"CARPENTIE

RE IN METALLO"},{"ateco_code":"16.23.2","ateco_description":"CARPENTIERE IN LEGNO"},

{"ateco_code":"43.99.0","ateco_description":"CARPENTIERE EDILE"},

{"ateco_code":"25.11.0","ateco_description":"CARPENTIERE IN FERRO"},

{"ateco_code":"25.11.0","ateco_description":"MONTAGGIO CARPENTIERE INDUSTRIALI"},

{"ateco_code":"25.11.0","ateco_description":"CARPENTIERE FABBRO"}]}

{"search_string":""PRODUZIONE"","risultato":"unici","0":[{"ateco_code":"n.c. ","ateco_description":"PRODUZIONE"}]}

{"search_string":""IMPIEGATO"","risultato":"falliti","0":[]}

L'output così prodotto in CIRCE tramite R è re-diretto da stdout verso rApache e quindi risale verso brew e viene incluso nella http response, per ritornare a WITCH.

WITCH a sua volta, a seguito di ulteriori check e manipolazioni ed eventuali segnalazioni al front-end, inoltra l'output verso il front-end affinché venga reso disponibile all'utente.

La struttura appena descritta viene definita *onion style execution*, ossia lancio a cascata dei vari moduli e ritorno e consente di supportare la complessità ed eterogeneità dell'architettura descritta.

4.3. Meccanismi di sicurezza

Oltre a quanto già esposto in merito alla validazione/bonifica dell'input utente ed al fatto di lavorare esclusivamente in memoria evitando scritture su filesystem sono state implementate le seguenti misure da adottare in esercizio:

- url rewriting (*security policy F*): Apache configurato per riscrivere le url attraverso l'uso del modulo mod_rewrite in modo tale che qualunque richiesta verso rApache punti sempre alla stessa pagina html in brewing R (e non a script o pagine differenti);
- accettare solo un ip specifico per le richieste (*security policy G*) : nella configurazione virtual host dell'application server sarà inserita una clausola 'allow from' per consentire solo ad uno specifico indirizzo ip (quello della macchina su cui gira WITCH) di poter effettuare richieste;
- static routes (*security policy H*, opzionale): consentire, una volta in esercizio, la raggiungibilità dell'application server solo attraverso routing statici.

4.4. Trattamento dell'output CIRCE in WITCH e gestione errori di codifica

WITCH effettua una serie di controlli e manipolazioni sull'output prodotto in CIRCE al fine di produrre un output conforme alle specifiche richieste.

I controlli e le manipolazioni sull'output mirano ad ottenere un formato UTF-8 privo di errori e tale da poter essere inviato al front-end per le successive operazioni. Il trattamento degli errori lato application server operato da WITCH è relativo alla eventuale propagazione verso il front-end di messaggi di errore opportunamente codificati. L'applicazione dispone di 3 livelli configurabili di segnalazione degli errori:

1. ***silent***, nessuna segnalazione;
2. ***text***, segnalazione in modalità testuale;
3. ***http status,*** segnalazione soap con status codes http.

I primi due riguardano modalità utili in sviluppo/debug (es. creare messaggi personalizzati in caso di errore); la terza è quella adottata in esercizio, che si compone di messaggi SOAP con status code http da ritornare al front-end in caso di situazioni anomale.

Il front-end riceve, in formato UTF-8, la risposta http e presenta i dati all'utente; in caso di errori può effettuare un check sulla segnalazione fornita dal back-end e ritornare all'utente una pagina con messaggi comprensibili di segnalazione del problema.

4.5. Supporto per analisi di qualità ed integrazione DBMS

Pur essendo tutta l'applicazione progettata come stateless, WITCH effettua delle scritture su database (***security policy I***) allo scopo di salvare le stringhe di ricerca immesse nel sistema. Il database, con charset UTF-8, è ospitato presso il DBMS MySql di esercizio.

Il software effettua le seguenti operazioni:

- copia di ogni stringa di ricerca propagata verso CIRCE, con relativo timestamp;

- invio settimanale, tramite email, delle stringhe immesse nell'ultima settimana (ogni lunedì);
- cancellazione giornaliera dei record vecchi più di 90 giorni (storico limitato a 3 mesi).

Alcune di queste operazioni sono svolte sul back-end tramite script schedulati (*crontab*) che accedono al DBMS e recuperano/cancellano i record in questione. Inoltre, tutte le operazioni di scrittura sono svolte prevenendo ipotetiche *sql injection* tramite *query* parametriche e bounded, anche se le operazioni avvengono esclusivamente lato back-end.

Le email contenenti le *query* utente sono inviate ogni settimana ad una lista di destinatari che, si occupano di controllare e migliorare l'efficacia del sistema di codifica.

La rimozione dei record oltre i 90 giorni è adottata allo scopo di non appesantire il database di produzione; il volume previsto di ricerche sfiora indicativamente per eccesso un milione all'anno (meno di 20.000 a settimana).

4.6. Considerazioni sulla sicurezza

Le policy di sicurezza previste (descritte in policy B,C,D,E,F,G,H,I nel corso delle pagine precedenti) sono state implementate per garantire un alto livello di sicurezza ed affidabilità a fronte del fatto che, ad esempio, rApache e brew consentirebbero, se non opportunamente reindirizzati, di avviare tramite http uno script R generico (o altro potenzialmente rischioso), oppure eseguire, con diritti dell'utente "apache", alcune chiamate di sistema intrinsecamente rischiose (es. exec).

Lo scopo è prevenire qualunque prevedibile metodica di utilizzo del sistema per ottenere privilegi superiori (privilege escalation) o produrre disservizi (tramite Denial of Service o tipologie di attacco mirato).

Di seguito un riepilogo ed una descrizione sintetica dello scopo delle security policies utilizzate:

- Policy B: previene all'origine il passaggio di stringhe con rischi di command/code *injection* (questo è garantito da Yii e dal custom validator).
- Policy C: previene il richiamo di system call (es. exec) intrinsecamente rischiose o il richiamo di comandi del SO o , ancora, la creazione di processi indipendenti
- Policy D,E: per prevenire scritture su filesystem, sia lato application server che lato Yii-back-end/WITCH;
- Policy F: url rewriting, per prevenire richieste arbitrarie sull'application server e forzare solo l'esecuzione dello script corretto;
- Policy G: ip filtering, per fare in modo che le richieste http all'application server pervengano solo da macchine autorizzate;
- Policy H: static routes, per fare in modo che la macchina dell'application server sia raggiungibile sono tramite un route predefinito e statico;
- Policy I: per mantenere dati persistenti (log delle ricerche) per analisi di qualità senza compromettere il blocco delle scritture su filesystem lato back-end ed application server, pur mantenendo una logica applicativa stateless.

In sintesi le policy adottate consentono di prevenire numerose categorie di attacchi riducendo le possibilità per un potenziale *attacker* di riuscire ad individuare e sfruttare eventuali vulnerabilità presenti nel sistema nel suo complesso, dunque di aumentare notevolmente la sicurezza a fronte dell'utilizzo di R come linguaggio per l'engine per la codifica Ateco (CIRCE).

4.7. Efficacia, efficienza e performance

Come già scritto nei paragrafi precedenti l'applicazione è strutturalmente ed intrinsecamente CPU intensive; l'esecuzione contemporanea di interpreti R in parallelo è la norma per un sistema di questo tipo. La gestione del multithreading è affidata allo stesso web server evitando di introdurre una eccessiva complessità a livello di codice applicativo (si consideri inoltre che il php non supporta il multithreading in modo nativo).

La soluzione adottata, realtiva all'*onion style execution*, consente di supportare la complessità connessa allo specifico problema di integrazione: utilizzare un software di codifica realizzato con

R, tipicamente un linguaggio destinato ad elaborazioni statistiche/scientifiche e non web-oriented, e di risolvere numerose problematiche relative all'efficienza ed alla sicurezza che altrimenti avrebbero costituito limiti assoluti alla possibilità concreta di utilizzare il software via web, specie dovendo esporre il servizio di codifica su Internet.

In termini prestazionali il sistema dovrebbe risultare più efficiente rispetto ad una soluzione (comunque non implementabile in esercizio) che preveda la creazione di processi serializzati separati in cui far operare l'interprete R, dunque vantaggioso anche in termini di performance e scalabilità futura.

Da test effettuati in previsione del rilascio dell'applicazione è emerso che in funzione della complessità della *query* sottoposta al sistema l'esecuzione delle sezioni interpretate sull'application server comporta circa dall'85% al 93% del tempo complessivo che va dall'invio della richiesta all'ottenimento della risposta restituita al front-end; le sezioni php sembrano supportare pienamente i requisiti di performance, probabilmente a causa dell'assenza di loop di durata significativa.

Una possibile strategia di ottimizzazione consisterebbe nell'individuare e modificare sezioni critiche di codice ove maggiore è la latenza (e dunque consumo di CPU), onde ridurre sotto stress il fenomeno di accodamento dovuto allo scheduling operato dal SO e conseguente tendenza alla saturazione.

I test di carico hanno indicato che sotto carichi elevati vi è una tendenza alla saturazione delle CPU nel back-end/application server, si rimanda al capitolo 5 sui test di carico per maggiori dettagli in merito.

In sintesi, il sistema web service WITCH, per le sue caratteristiche oltre a garantire la massima efficienza, efficacia e sicurezza (in linea con gli standard previsti dall'Istituto) nell'attuale impiego per la codifica delle attività economiche, è un sistema (ri)utilizzabile in futuro in applicativi che necessiteranno una codifica on-line di variabili testuali.

Un possibile utilizzo potrebbe essere previsto nei questionari elettronici sviluppati per la fase di acquisizione di un'indagine oppure per la codifica di variabili per le quali l'Istat deve rilasciare dati secondo classificazioni ufficiali di riferimento, come la "professione" o il "titolo di studio", come descritto nel capitolo 7.

5. Test di sicurezza e di carico

L'applicazione di codifica on-line per l'ATECO 2007 è stata sottoposta a test di sicurezza e di carico per individuare ed eliminare i punti di vulnerabilità applicativa e per valutare la qualità della risposta e la stabilità del sistema in condizioni di utilizzo variabili.

5.1 Test di sicurezza

La gestione della sicurezza è di vitale importanza per un'applicazione web. Infatti, secondo la fonte Gartner oltre il 75% degli attacchi vengono indirizzati direttamente alle applicazioni, causando gravi danni di immagine e pesanti perdite finanziarie.

Gli obiettivi degli attacchi sono le vulnerabilità che si celano all'interno delle applicazioni software. Le vulnerabilità applicative sono quasi sempre presenti: fino ad ora le politiche di qualità del software ed i relativi investimenti si sono concentrati sulla correzione dei difetti funzionali e sulle performance trascurando la sicurezza del codice prodotto. Secondo Microsoft Developer Research, alcuni analisti affermano che il 64% degli sviluppatori non sono confidenti di poter scrivere applicazioni sicure. Ciò comporta l'introduzione di possibili vulnerabilità software da parte degli stessi team di sviluppo interni alle aziende, da parte dei fornitori IT o dall'uso di soluzioni open.

Il numero di tipologie di attacco è in costante crescita e come cita il rapporto Clusit 2015[13]: "in questa fase storica la superficie di attacco complessivamente esposta dalla nostra civiltà digitale cresce più velocemente della nostra capacità di proteggerla". Da un'analisi degli incidenti informa-

[13] http://clusit.it/rapportoclusit/

tici più significativi degli ultimi 12 mesi messi a confronto con i 36 mesi precedenti, si evince appunto una sensibile evoluzione degli scenari d'attacco. Come è possibile evincere dal rapporto Verizon[14], nel 2015 sono stati rilevati 79.790 incidenti sulla sicurezza informatica di cui 2.122 relativi a Data Loss.

Soffermandoci sul conteggio delle vulnerabilità pubblicate negli ultimi 10 anni (dal 2006 al 2015), secondo il National Vulnerability Database[15] del NIST (dati aggiornati al 29 dicembre 2015),è possibile notare che il 2015 vede una diminuzione del numero di CVE pubblicati (Common Vulnerabilities and Exposures) di circa un 20% rispetto al 2014 (anno caratterizzato dal massimo numero di record generati). Ad ogni modo, se consideriamo il CVSS (Common Vulnerability Scoring System) che rappresenta uno dei migliori indicatori della natura e dell'impatto potenziale di qualsiasi vulnerabilità ed utilizzato dal NIST per il calcolo del Base Score da applicare ad ogni vulnerabilità analizzata e confermata, si nota che il 2015, rispetto all'anno 2014, presenta un aumento dal 13% al 22% circa per i CVE con con valore CVSS superiore al 7.5 (quindi ad alto impatto). Allo scopo di limitare i tentativi di attacco applicativo al sistema è stata avviata una attività di ottimizzazione (in ambito sicurezza) sia a livello infrastrutturale sia a livello applicativo.

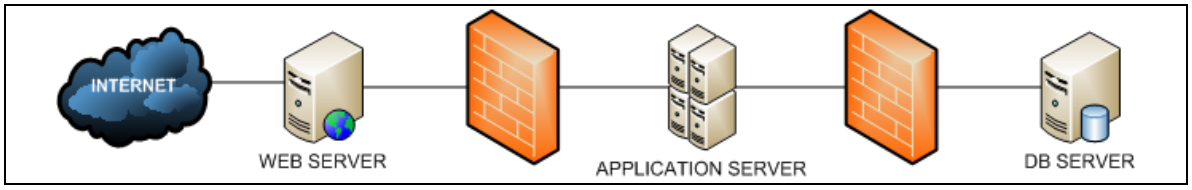
L'architettura utilizzata prevede la tipica suddivisione in layer: N-Tier/3-Tier[16]. Tale suddivisione si rivela una soluzione vincente sia nell'ottica della limitazione di accoppiamento e di duplicazione funzionale sia nell'ambito della sicurezza applicativa. Infatti suddividere l'applicazione in più parti (logiche e fisiche) permette di limitare la superficie di attacco e la sensibilità del sistema a vulnerabilità o a bug legati all'applicazione e all'ambiente in cui la stessa viene eseguita.

Nell'ottica di quanto introdotto, il sistema si suddivide nei seguenti layer:

1. Web server;
2. Application Content;
3. Datastore.

Ciascuno layer è separato da un sistema di sicurezza perimetrale (Firewall) in grado di limitarne la comunicazione su specifici socket (canale di comunicazione fra due processi in rete), come rappresentato, a titolo esemplificativo, dalla figura seguente.

Figura 14 - Esempio Architettura 3-Tier



Prima di procedere con la descrizione di come è stato gestito l'aspetto della sicurezza per l'applicazione WITCH-CIRCE è bene specificare la differenza sostanziale tra i termini "attacchi" e "vulnerabilità" che spesso vengono confusi. In particolare, l'attacco è la tecnica utilizzata dal malintenzionato per sfruttare le vulnerabilità presenti in un'applicazione o in un sistema; la vulnerabilità è invece una debolezza legata ad un possibile errore di progettazione applicativa/infrastrutturale o relativo ad un bug implementativo dell'applicazione stessa.

Allo scopo di limitare le vulnerabilità del sistema è stata effettuata una attività di *penetration* test (PenTest) eseguendo una serie di test approfonditi in modalità Ethical Hacking. L'attività, effettuata con strumenti automatici e manuali, si è basata su tecniche di attacco inferenziale ed ha avuto lo scopo di identificare l'eventuale presenza di vulnerabilità.

Le tecniche di PenTest, dunque, si basano sulla simulazione delle operazioni comunemente ese-

[14] http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigation-report-2015_en_xg.pdf
[15] https://nvd.nist.gov/
[16] Nell'ingegneria del software, il termine architettura multi-Tier o n-Tier (dall'inglese multi-Tier architecture, un'architettura multi-strato) indica un'architettura software in cui le varie funzionalità del software sono logicamente separate ovvero suddivise su più strati o livelli software differenti in comunicazione tra loro. 3-Tier è un'architettura client-server in cui l'interfaccia utente, i processi logico funzionali, l'archiviazione informatica dei dati e l'accesso ai dati sono sviluppate e mantenute come moduli indipendenti.

guite da un agente di minaccia esterno o interno e fanno uso di strumenti e tecniche tipicamente adoperate in contesti di reale attacco ai sistemi.

L'approccio utilizzato per le attività di test è stato sia di tipo black-box che white-box cioè, rispettivamente, effettuando test senza conoscenza dei dettagli implementativi ed effettuando test condividendo con il team di sviluppo il dettaglio relativo a processi e flussi applicativi.

La normale attività di pentesting è stata effettuata in base ai seguenti passi:

1. ricerca di informazioni;
2. ricerca di vulnerabilità:
    a. test applicazione;
    b. test web server;
3. stesura del report;
4. supporto per la correzione delle vulnerabilità riscontrate.

L'attività di ricerca informazioni è stata effettuata sia tramite Web-Spider sia tramite User Direct Spider allo scopo di:

- Rintracciare contenuti nascosti:
    - file/directory;
    - informazioni utili;
    - eventuali interfacce di back end.
- Cercare informazioni sul web server:
    - versioni per eventuali bug;
    - presenza funzionalità di default;
    - http fingerprinting / banner grabbing.
- Cercare informazioni sullo sviluppo:
    - nomi file o directory;
    - Informazioni codice sorgente.
- Analizzare e gestire gli errori;
- Tracciare le funzionalità dinamiche;
- Effettuare il mapping del web service.

L'attività di ricerca delle vulnerabilità è stata effettuata testando:

- la presenza di meccanismi di autenticazione/autorizzazione;
- la gestione di eventuali sessioni;
- la validazione degli input (con apposito controllo in merito ad eventuale presenza di meccanismi in grado eseguire script malevoli invocando apposite funzioni "R");
- la logica applicativa;
- l'accesso a dati;
- l'information disclosure;
- i controlli client side (per eventuali attacchi al client);
- gli attacchi all'infrastruttura.

L'applicazione di codifica on-line per l'ATECO 2007 è stata sottoposta a test di sicurezza, tramite i seguenti strumenti:

- Nessus (controllo vulnerabilità architettura e applicative);
- Nikto (controllo vulnerabilità architettura e applicative);
- BURP Suite Pro (controllo vulnerabilità applicativa);
- OWASP Zed Attack Proxy Project (controllo vulnerabilità applicativa);
- test di tipo manuale (BackEnd/FrontEnd).

È importante evidenziare che sia in fase di progettazione sia in fase di sviluppo dell'applicazione/architettura applicativa è stata posta enorme attenzione sul rispetto degli standard OWASP (Open *Web application* Security Project) ed in particolar modo sulle metodologie di sviluppo da utilizzare per mitigare ed infine eliminare eventuali vulnerabilità.

L'utilizzo di appositi frame work di sviluppo (Yii) ha permesso inoltre di rispettare gli standard

di sicurezza alla base di un SDLC (System Development Life Cycle) sicuro.

L'unione delle diverse metodologie e tecnologie di test ha permesso l'eliminazione di numerose problematiche sorte durante la fase di codifica del sistema. In particolare, quelle più evidenti in fase iniziale sono raggruppabili in due tipologie:

1. Problematiche Applicative.
    a. XSS (Cross-Site Scriptiong): permette di inserire o eseguire codice lato client al fine di attuare un insieme variegato di attacchi;
    b. CSRF (Cross-Site Request Forgery): il sistema riceve richieste da un client senza meccanismi di controllo sulla intenzionalità della richiesta stessa;
    c. ClickJacking: permette di anteporre degli iframe a vari livelli consentendo "navigazioni" non convenzionali;
    d. INPUT vulnerabile con accesso al codice R[17].
2. Problematiche Web Server.
    a. HTTP Security Headers non configurati opportunamente;
    b. codifica caratteri non gestita.

I report generati classificano le problematiche assegnando una scala gerarchica di gravità: High, Medium, Low, Information. Le informazioni ottenute dai vari sistemi (automatici e manuali) sono state correlate allo scopo di validare quanto riportato e di rilevare la presenza di falsi positivi o falsi negativi.

Seguono alcuni estratti dei report generati durante la fase di PenTest. La figura 15 è relativa alla sezione iniziale del report sulle vulnerabilità applicative prodotto dal software ZAP OWASP; è possibile osservare la variazione nel numero di vulnerabilità riscontrate durante la prima scansione

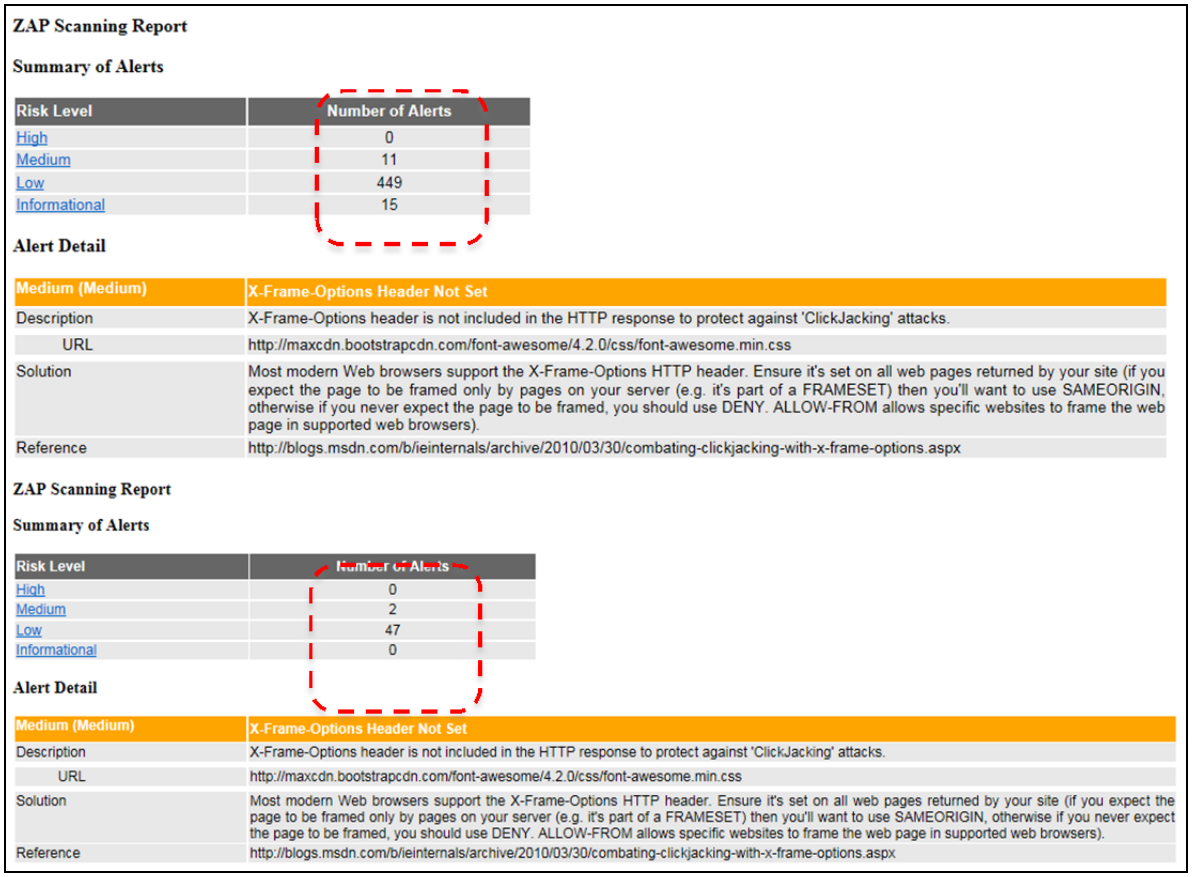
- 0 vulnerabilità di tipo "High";
- 11 vulnerabilità di tipo "Medium";
- 449 vulnerabilità di tipo "Low";
- 15 vulnerabilità di tipo "Informational".

e le vulnerabilità presenti dopo le opportune correzioni :

- 0 vulnerabilità di tipo "High"
- 2 vulnerabilità di tipo "Medium"
- 47 vulnerabilità di tipo "Low"
- 0 vulnerabilità di tipo "Informational"

[17] The RAppArmor Package: Enforcing Security Policies in R Using Dynamic Sandboxing on Linux - Jeroen Ooms [UCLA Department of Statistics]

Figura 15 - Estratto Report ZAP OWASP



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	11
Low	449
Informational	15

Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx

ZAP Scanning Report

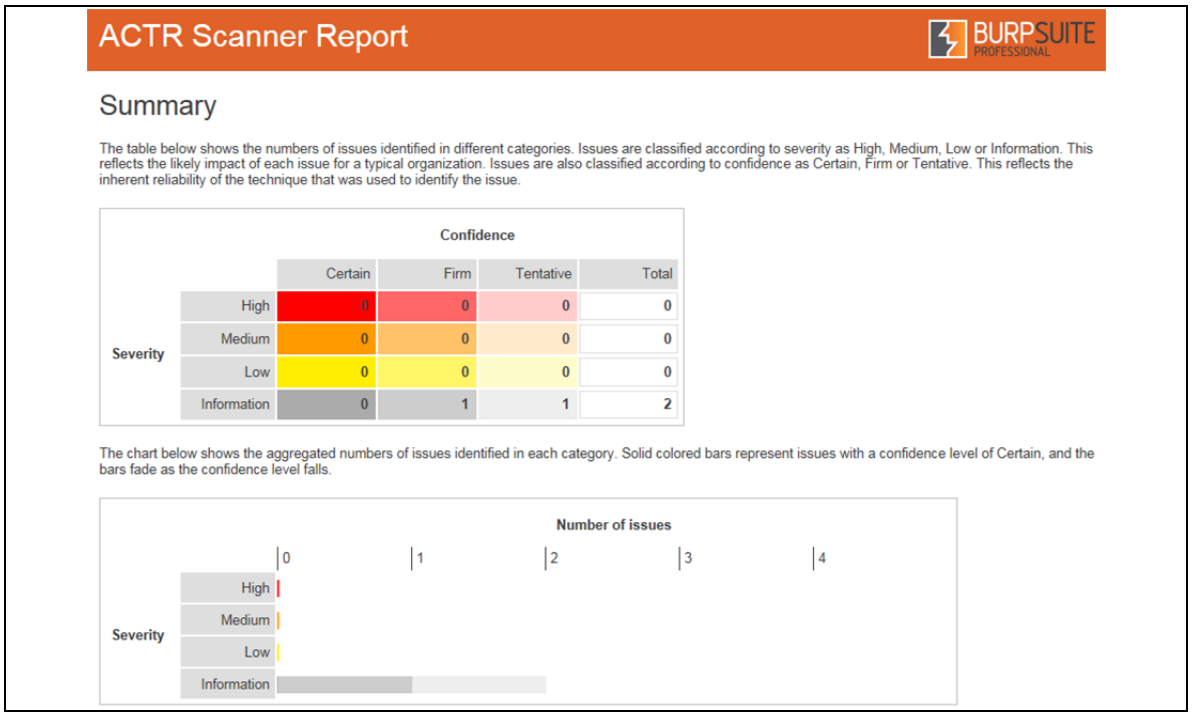
Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	2
Low	47
Informational	0

Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx

Figura 16 - Estratto Report BURP Suite Professional



ACTR Scanner Report

BURPSUITE PROFESSIONAL

Summary

The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low or Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

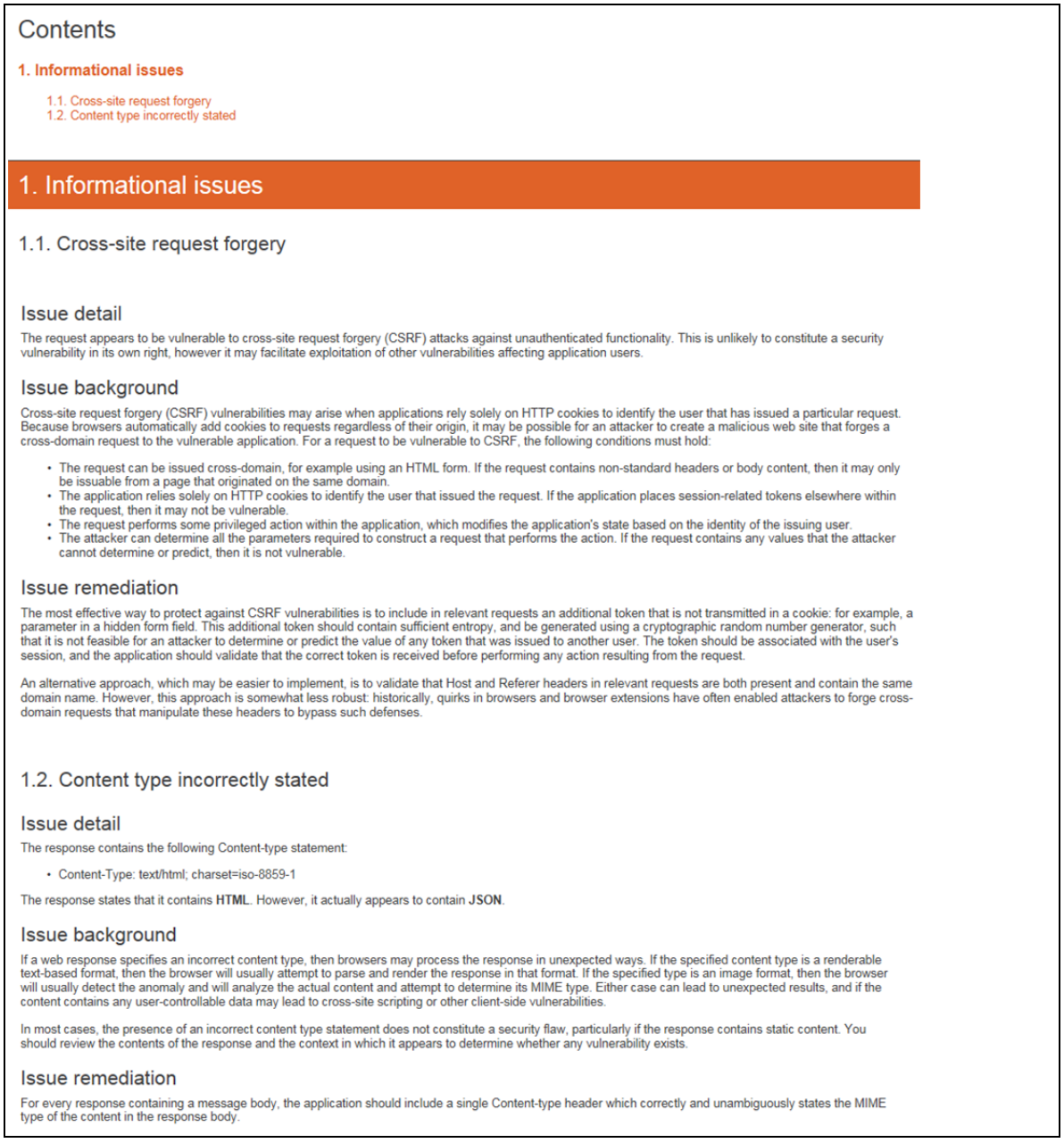
Severity \ Confidence	Certain	Firm	Tentative	Total
High	0	0	0	0
Medium	0	0	0	0
Low	0	0	0	0
Information	0	1	1	2

The chart below shows the aggregated numbers of issues identified in each category. Solid colored bars represent issues with a confidence level of Certain, and the bars fade as the confidence level falls.

In figura 16 si mostra un riepilogo delle vulnerabilità riscontrate durante i test effettuati tramite un ulteriore software di controllo: BURP Suite Professional. Segue (figura 16) il report che espone

la descrizione puntuale delle due vulnerabilità di tipo "Information" e le eventuali tecniche di "Remediation" utili agli sviluppatori per la risoluzione/mitigazione delle problematica di sicurezza applicativa riscontrate.

Figura 17 - Dettaglio Report BURP Suite Professional



Contents

1. Informational issues

1.1. Cross-site request forgery
1.2. Content type incorrectly stated

1. Informational issues

1.1. Cross-site request forgery

Issue detail

The request appears to be vulnerable to cross-site request forgery (CSRF) attacks against unauthenticated functionality. This is unlikely to constitute a security vulnerability in its own right, however it may facilitate exploitation of other vulnerabilities affecting application users.

Issue background

Cross-site request forgery (CSRF) vulnerabilities may arise when applications rely solely on HTTP cookies to identify the user that has issued a particular request. Because browsers automatically add cookies to requests regardless of their origin, it may be possible for an attacker to create a malicious web site that forges a cross-domain request to the vulnerable application. For a request to be vulnerable to CSRF, the following conditions must hold:

- The request can be issued cross-domain, for example using an HTML form. If the request contains non-standard headers or body content, then it may only be issuable from a page that originated on the same domain.
- The application relies solely on HTTP cookies to identify the user that issued the request. If the application places session-related tokens elsewhere within the request, then it may not be vulnerable.
- The request performs some privileged action within the application, which modifies the application's state based on the identity of the issuing user.
- The attacker can determine all the parameters required to construct a request that performs the action. If the request contains any values that the attacker cannot determine or predict, then it is not vulnerable.

Issue remediation

The most effective way to protect against CSRF vulnerabilities is to include in relevant requests an additional token that is not transmitted in a cookie: for example, a parameter in a hidden form field. This additional token should contain sufficient entropy, and be generated using a cryptographic random number generator, such that it is not feasible for an attacker to determine or predict the value of any token that was issued to another user. The token should be associated with the user's session, and the application should validate that the correct token is received before performing any action resulting from the request.

An alternative approach, which may be easier to implement, is to validate that Host and Referer headers in relevant requests are both present and contain the same domain name. However, this approach is somewhat less robust: historically, quirks in browsers and browser extensions have often enabled attackers to forge cross-domain requests that manipulate these headers to bypass such defenses.

1.2. Content type incorrectly stated

Issue detail

The response contains the following Content-type statement:

- Content-Type: text/html; charset=iso-8859-1

The response states that it contains **HTML**. However, it actually appears to contain **JSON**.

Issue background

If a web response specifies an incorrect content type, then browsers may process the response in unexpected ways. If the specified content type is a renderable text-based format, then the browser will usually attempt to parse and render the response in that format. If the specified type is an image format, then the browser will usually detect the anomaly and will analyze the actual content and attempt to determine its MIME type. Either case can lead to unexpected results, and if the content contains any user-controllable data may lead to cross-site scripting or other client-side vulnerabilities.

In most cases, the presence of an incorrect content type statement does not constitute a security flaw, particularly if the response contains static content. You should review the contents of the response and the context in which it appears to determine whether any vulnerability exists.

Issue remediation

For every response containing a message body, the application should include a single Content-type header which correctly and unambiguously states the MIME type of the content in the response body.

Concludendo, a seguito dei test e dei correttivi apportatati, l'applicazione di codifica, che peraltro non ha mai mostrato rischi di alta entità, anche grazie alle scelte architetturali seguite ed ai livelli di sicurezza introdotti dai frame work di sviluppo, si può considerare sicura da attacchi esterni attualmente conosciuti.

5.2 Test di carico

L'applicazione di codifica web è stata sottoposta a test di carico, finalizzati a valutare la qualità della risposta ed i limiti di resistenza in condizioni di carico eccezionali.

Per il test di carico è stato utilizzato il prodotto Jmeter. Si tratta di un tool java open source svi-

luppato dall'Apache Software Foundation. Il prodotto Jmeter è stato affiancato, in fase di report, dalle statistiche effettuate con il prodotto di monitoraggio Splunk Enterprise.

Con lo strumento di test Jmeter viene simulata la navigazione sul sito effettuata da un numero preordinato di visitatori. Ogni visitatore esegue una sequenza prestabilita di operazioni per un numero limitato di volte. Tale numero può essere definito da un limite di tempo o da una quantità massima di cicli.

Nel caso in esame, le operazioni eseguite da ciascun utente consistono nell'inserimento di una stringa di ricerca nel modulo predisposto e nel conseguente invio della richiesta al server. Più in dettaglio, si tratta di misurare i tempi di risposta della pagina che esegue la seconda parte della procedura e impegna il server applicativo nelle modalità descritte in dettaglio nel capitolo 4.

Contemporaneamente all'esecuzione dei test, con Splunk è stato possibile tenere sotto controllo alcune variabili di sistema del server applicativo insieme ai tempi di risposta registrati sul server. Questo prodotto si è rivelato molto utile per integrare i dati raccolti con il programma Jmeter. Configurando opportunamente Splunk si possono reperire, infatti, tutti i dati di funzionamento della macchina. Il risultato può poi essere rappresentato graficamente in modo semplice ed efficace. Il tutto analizzando ed interpretando file di testo.

Nella pagine che seguono, insieme ai report del simulatore Jmeter, vengono mostrati i grafici ottenuti con Splunk analizzando i file di log di Apache ed i file delle statistiche di sistema prodotti dal server applicativo.

5.2.1Test di carico realizzati

Nel testo che segue, vengono riportati i risultati dei test di prestazione e di carico effettuati sul sito dell'Ateco 2007 l'11 giugno 2015.

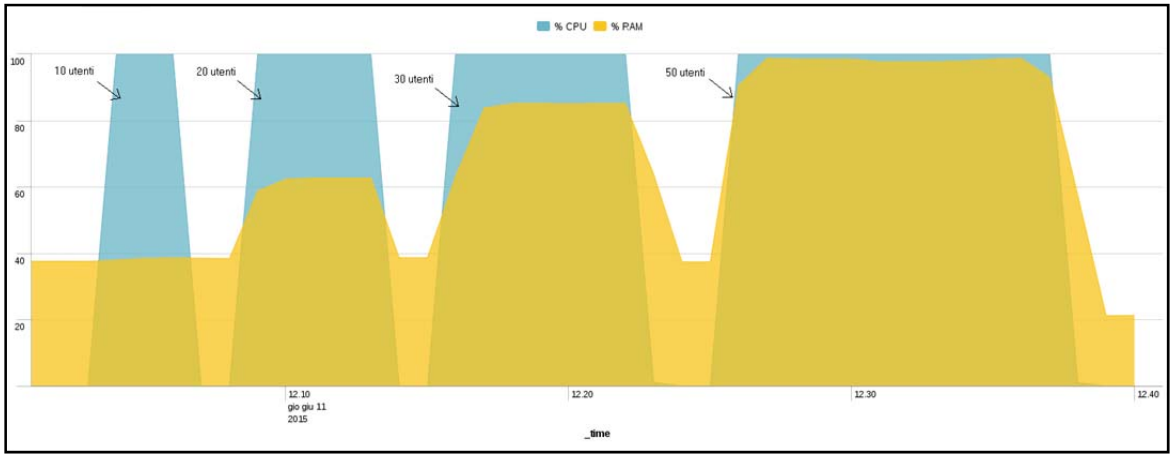
Un primo tipo di test (performance test) mette alla prova il sistema con diversi livelli di carico e consiste nel simulare la navigazione sul sito effettuata da un numero preordinato di visitatori concorrenti. Questi test non cercano direttamente un limite del server, ma servono piuttosto per provare la qualità della risposta e la stabilità del sistema in condizioni di utilizzo variabili.

Un altro tipo di test (stress test) è quello che mira ad individuare i limiti del sistema. Consiste nel caricare il server aumentando gradualmente il numero di utenti contemporanei fino a raggiungere un punto di cedimento, ovvero una condizione di saturazione delle risorse e di crisi del servizio.

Nel primo tipo di test vengono simulati 10, 20, 30 e 50 utenti contemporanei. Ogni utente esegue le sue richieste per 100 volte.

Nel grafico che segue viene mostrata le variazione nel tempo dell'utilizzo di memoria e dei processori (RAM e CPU) durante la sequenza di test.

Figura 18 - Grafico (Splunk) delle risorse di sistema

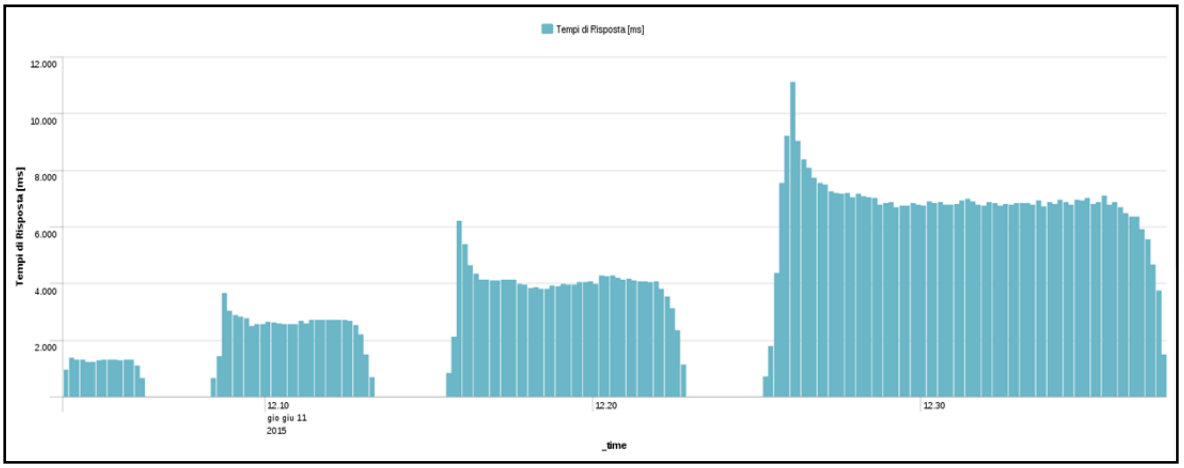


Dal grafico si vede che con 10 utenti concorrenti viene già raggiunto il 100% dell'utilizzo di CPU. Questo conferma il fatto, già accennato nel capitolo 4, che il processo di elaborazione fa un uso intensivo dei processori. Tuttavia, la saturazione della CPU non compromette direttamente

l'utilizzabilità del sito. L'evento determinante, nel produrre la crisi di sistema, è la saturazione della memoria RAM; come sarà evidente nel secondo tipo di test.

Per la stessa sequenza di test (10, 20, 30 e 50 utenti concorrenti), nella figura 19 vengono mostrati i tempi di risposta del sito al variare del tempo.

Figura 19 - Grafico (Splunk) dei tempi di risposta



Nei 2 grafici precedenti si vede come la durata di ciascun blocco di test aumenta in modo proporzionale al numero di utenti simulati. Questo perché ogni blocco di test termina quando è stato completato il numero totale delle richieste (Figura 18).

Di seguito vengono riportati 4 grafici che mostrano in un istogramma, la distribuzione dei tempi di risposta nei 4 casi (10, 20, 30 e 50 utenti concorrenti).

Figura 20 - Distribuzione dei tempi di risposta per 10 utenti

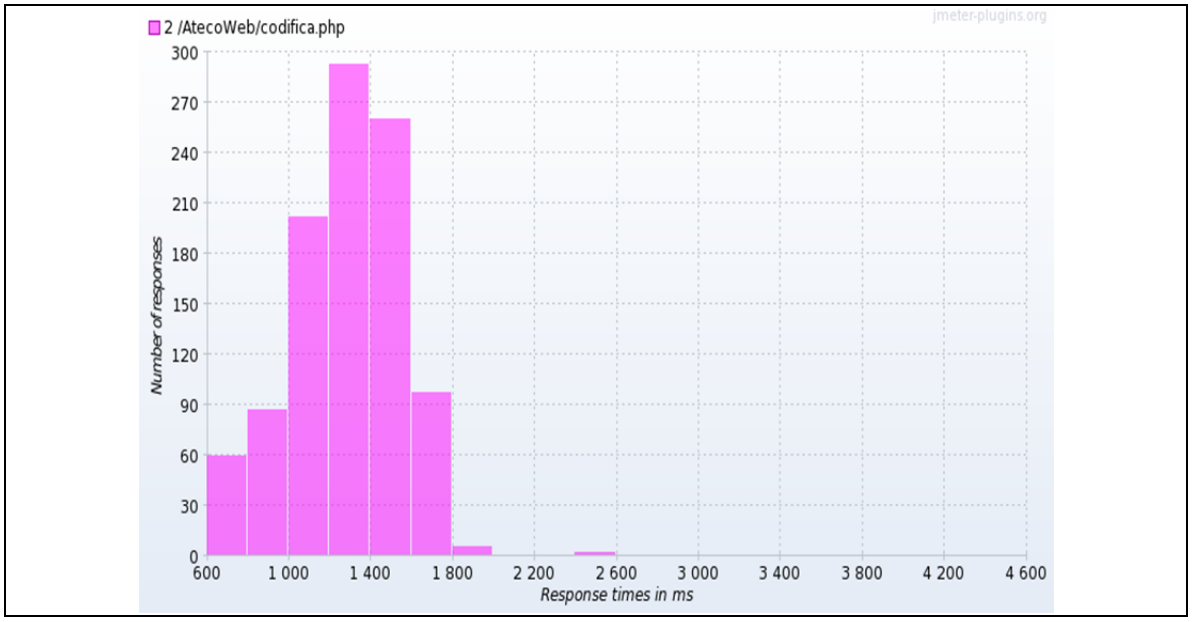


Figura 21 - Distribuzione dei tempi di risposta per 20 utenti

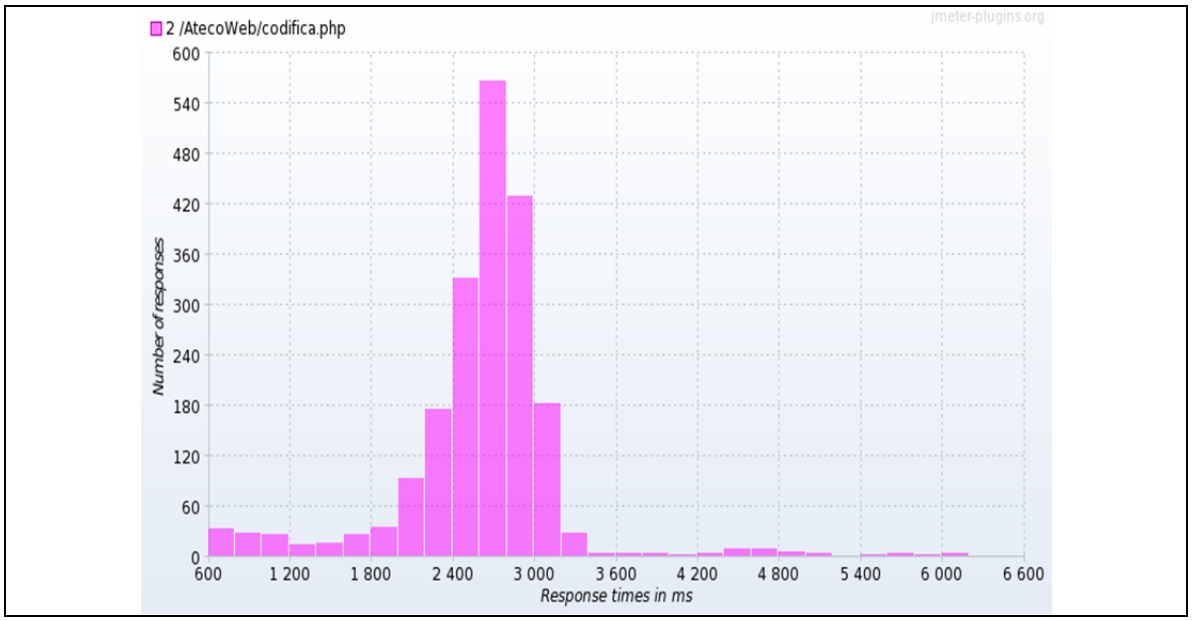


Figura 22 - Distribuzione dei tempi di risposta per 30 utenti

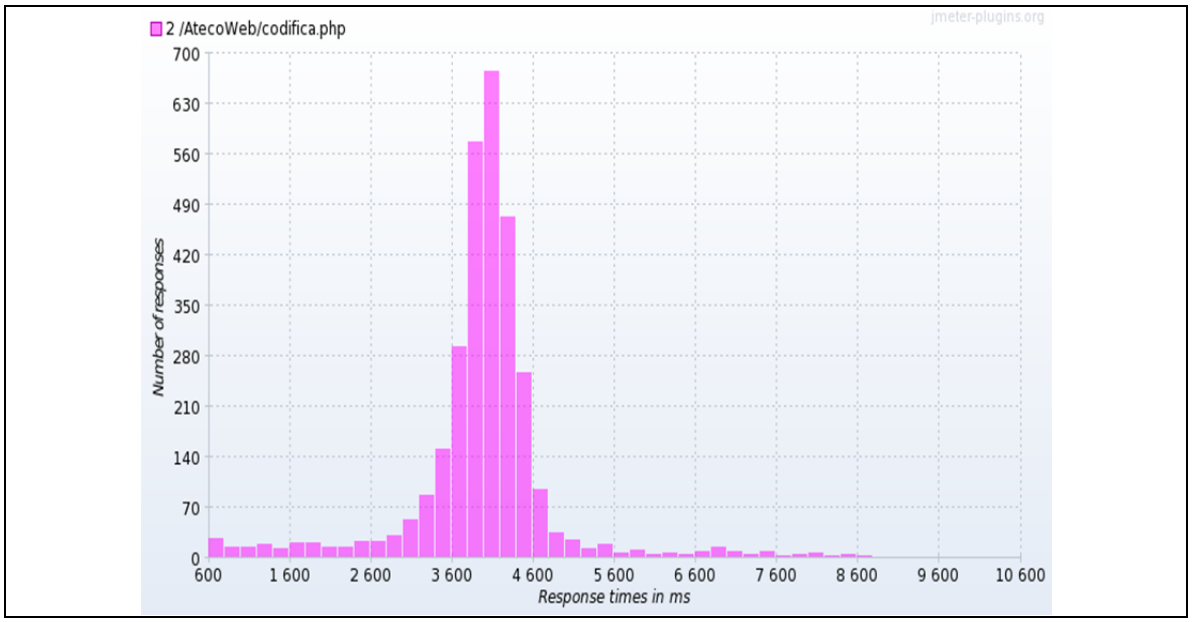
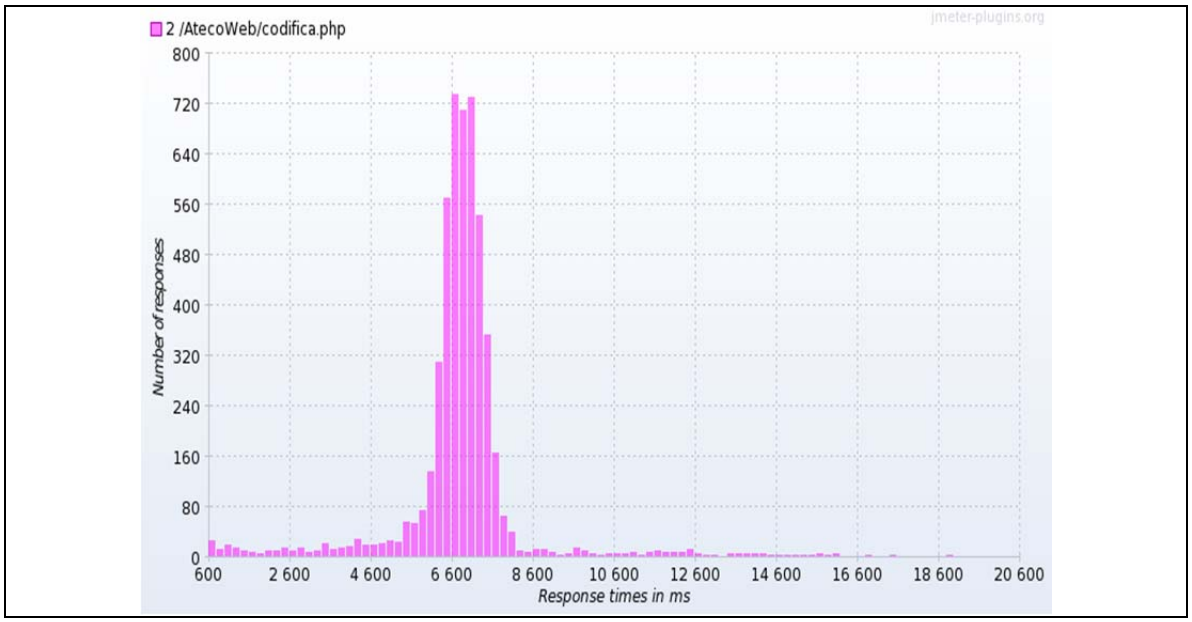


Figura 23 - Distribuzione dei tempi di risposta per 50 utenti



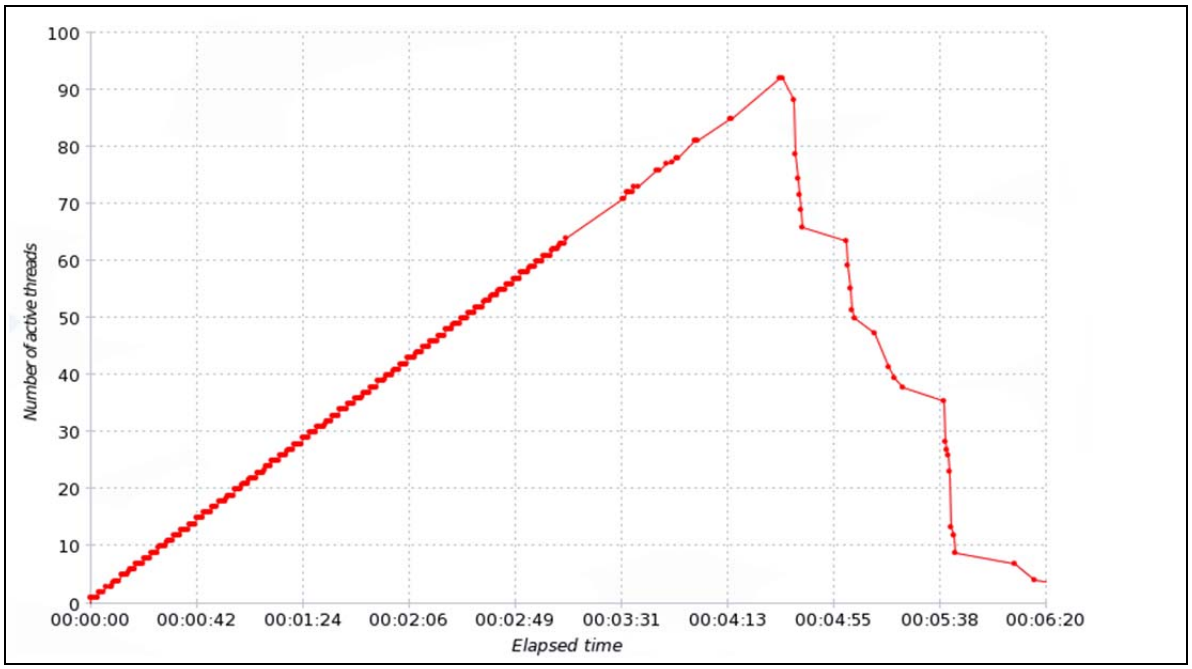
Nella tabella che segue sono esposti in sintesi i tempi medi di risposta ottenuti nei 4 casi di test. I valori mostrati in tabella sono ottenuti mediando tra 2 ripetizioni dello stesso test e sono sostanzialmente in linea con quelli della precedente applicazione web (basata su ACTR v3).

Tavola 20 - Tabella riassuntiva dei tempi di risposta

Utenti	10	20	30	50
Totale richieste	1.000	2.000	3.000	5.000
Tempo medio (millisecondi)	1.293	2.607	3.976	6.688
Errori %	0,00%	0,00%	0,00%	0,00%

Il secondo tipo di test mira ad individuare più direttamente i limiti del sistema. Si tratta di caricare il server fino a raggiungere un punto di cedimento, ovvero una condizione di saturazione delle risorse e di crisi del servizio. Per comprendere i risultati del test occorre leggere insieme i tre grafici seguenti, le cui ascisse mostrano la correlazione temporale tra il numero di utenti e il punto di crisi del sistema, come dettagliato nel seguito.

Figura 24 - Grafico nel tempo (Jmeter) dei thread attivi



Un test di questo tipo si può fare solamente su una macchina dedicata, identica o facilmente confrontabile con il server di produzione.

Il test consiste nel portare in attività sul sito 100 utenti contemporanei in un tempo transitorio di 300 secondi (figura 24).

Per ottenere il grafico di figura 25 è stato installato sul server sottoposto a test, uno strumento di Jmeter (ServerAgent) capace di inviare, durante il test, i dati di sistema (RAM, CPU,ecc.).

Nel grafico di figura 26 vengono mostrati i tempi di risposta misurati durante il test, riportando in ascissa il tempo trascorso dall'inizio del test ed in ordinata il tempo di risposta delle pagine espresso in millisecondi.

La rampa da 0 a 100 utenti si è interrotta prima dei 5 minuti previsti, in quanto al raggiungimento del limite il server inizia a non rispondere in modo controllabile. A quel punto il test può essere terminato.

Dopo 3 minuti circa, quando gli utenti (thread) hanno raggiunto quota 70 (figura 24), il server è entrato in crisi: l'utilizzo di memoria RAM è arrivato al 100%, ed è iniziato l'utilizzo pesante della scrittura su disco (I/O in figura 25). I tempi di risposta aumentano in modo non lineare e il sito (come il server) diventa inutilizzabile (figura 26). Si può quindi collocare intorno a 70, la quantità massima di utenti che il sistema può sopportare con la configurazione sottoposta a test.

Figura 25 - Grafico nel tempo (Jmeter) delle variabili di sistema durante la rampa di accessi

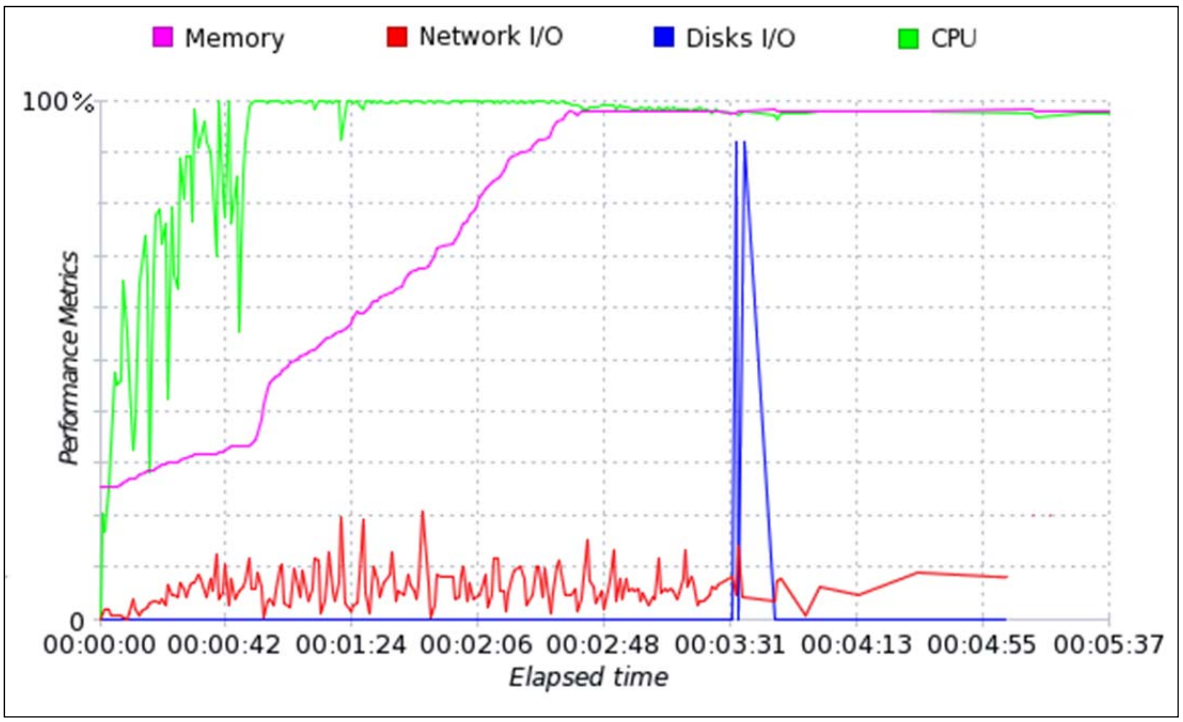
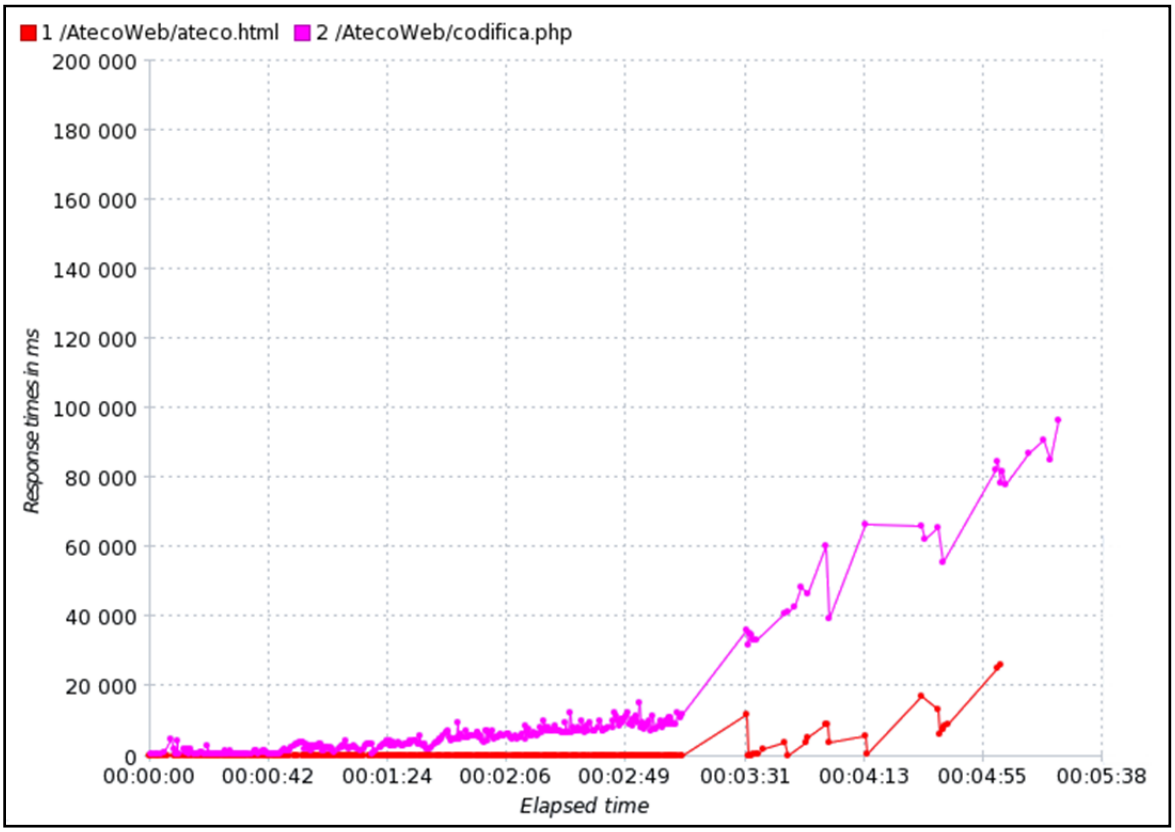


Figura 26 - Grafico nel tempo (Jmeter) dei tempi di risposta durante la rampa di accessi



5.2.2 Validità dei test di carico

Una corretta procedura di test risulta tanto più efficace quanto più la si associa, contestualmente, ad un meccanismo di "feedback" continuo con gli sviluppatori. Questa modalità operativa permette, infatti, di individuare e di risolvere gli eventuali malfunzionamenti rilevati nella fase di test.

Questo meccanismo coinvolge anche la configurazione di sistema: lavorando opportunamente sui parametri di funzionamento e sulle dotazioni hardware del server si può spostare in avanti il limite di utilizzo, arrivando a tollerare un maggior numero di utenti contemporanei. L'intervento sulla dotazione hardware del server è stato inoltre reso più agevole dal fatto che è stato utilizzato un server virtuale, che consente di modificare in tempo reale la dotazione di RAM o CPU ricorrendo solo in extremis al riavvio della macchina. Nel caso in esame si è passati da 1 a 4 processori e da 4 a 8 GB di RAM. I risultati mostrati nei grafici precedenti si riferiscono a questa ultima configurazione del server.

Il numero di utenti concorrenti simulato nei test di prestazione (10, 20, 30 e 50) è di un ordine di grandezza superiore rispetto alle quantità previste sulla base degli accessi della precedente applicazione. Un altro punto di riferimento è stato il sito istituzionale www.istat.it che fornisce l'accesso all'interfaccia utente: il numero di utenti attivi contemporanei varia intorno ad una media di 5 e raramente supera quota 20. Essendo l'applicazione Ateco solo uno dei servizi offerti dal sito istituzionale, questi valori possono essere considerati come estremi superiori.

I tempi di risposta ottenuti nelle varie prove a utenti fissi ed il limite ottenuto con il test a rottura sono da considerarsi ampiamente accettabili anche in vista degli sviluppi futuri nell'utilizzo dell'applicazione di codifica on-line. Un miglioramento ulteriore, sia dal punto di vista delle prestazioni che da quello delle garanzie sulla continuità di servizio, si potrà ottenere affiancando al server sottoposto a test un altro server con caratteristiche simili.

6. Il nuovo front-end per la ricerca on-line dei codici Ateco

La realizzazione del nuovo front-end per la consultazione, navigazione e codifica dell'Ateco 2007 era uno degli obiettivi del Piano di accessibilità 2014[18].

Tra novembre 2014 e marzo 2015 è stata dunque sviluppata una nuova applicazione per la consultazione on-line della classificazione Ateco 2007 e per la ricerca dei codici delle attività economiche.

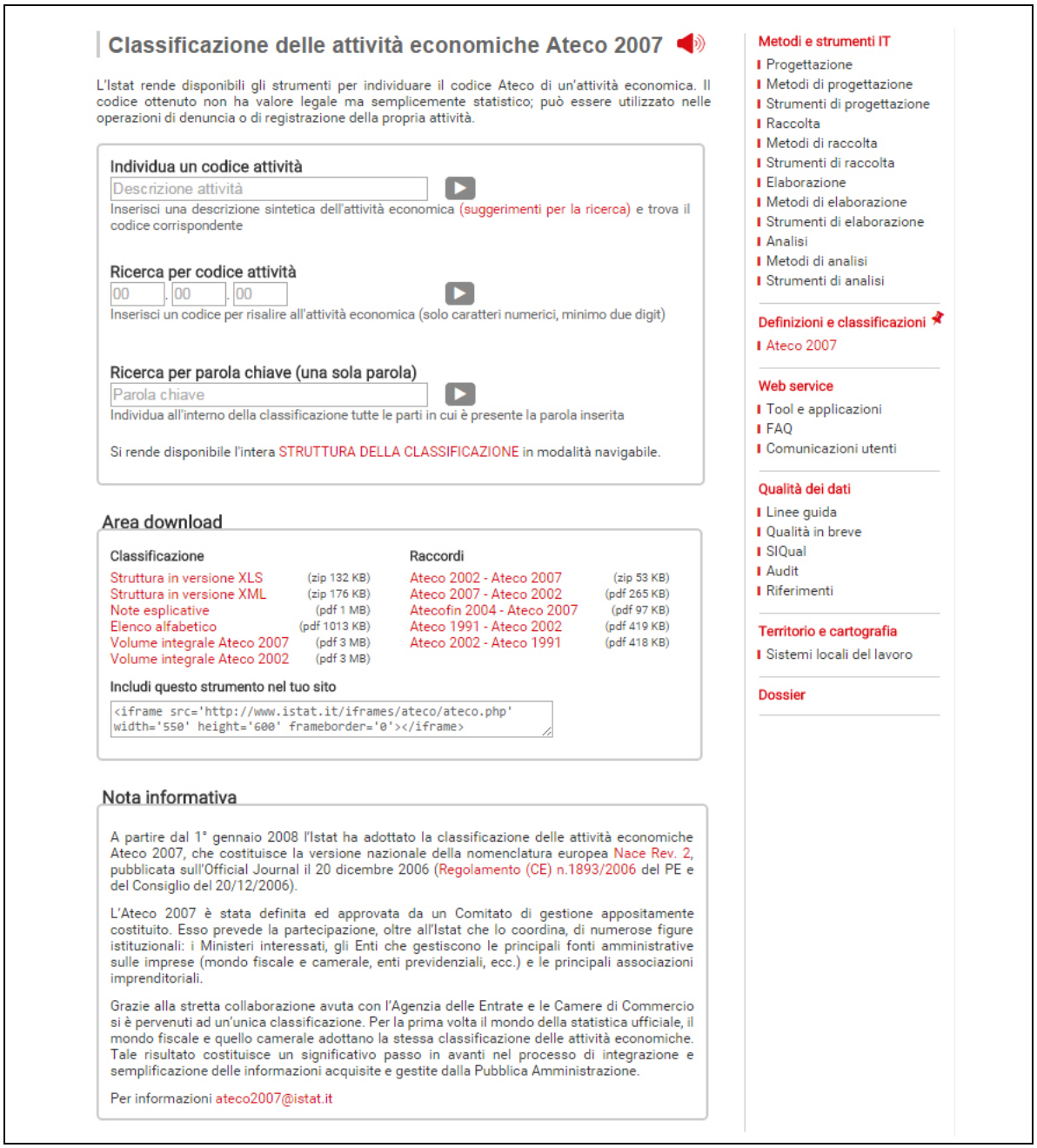
Le innovazioni realizzate sono numerose e comprendono:

- l'introduzione di nuove funzionalità interattive tramite l'uso di javascript e web services che permettono di ricavare l'attività economica partendo dal corrispondente codice Ateco a due, quattro o sei cifre;
- l'affinamento dello strumento di ricerca per parola chiave all'interno della classificazione;
- l'inserimento in uno stesso contesto degli strumenti di ricerca, codifica e consultazione dell'intera classificazione per offrire all'utente la possibilità di integrare i risultati ottenuti;
- la raccolta nella stessa cornice di tutti i file xml, xls e pdf relativi alla classificazione, in modo da facilitare l'uso delle funzionalità di ricerca e codifica dell'attività economica;
- la possibilità di inserire all'interno del proprio sito web, tramite iframe, tutto il front-end.

All'indirizzo www.istat.it/it/strumenti/definizioni-e-classificazioni/ateco-2007 è disponibile il risultato del lavoro (Figura 27).

[18] Il Piano di accessibilità annuale redatto ai sensi dell'articolo 7 del decreto legge n. 179 del 18 ottobre 2012 è disponibile sul sito web dell'Istat, all'indirizzo www.istat.it/it/accessibilità.

Figura 27 - Nuovo front-end per la consultazione dei codici Ateco 2007



6.1 Il progetto

Nel precedente sito Istat, i due sistemi di identificazione e ricerca dei codici delle attività economiche si trovavano in due sezioni distinte e separate ed erano accessibili da pagine diverse.

Il primo sistema riguardava l'applicazione di codifica del testo digitato attraverso il sistema ACTR v3 ampiamente descritta nei capitoli precedenti. Con il nuovo front-end si è colta l'occasione per un'ottimizzazione del codice in modo da rendere più semplice il passaggio dal software ACTR v3 al nuovo WITCH-CIRCE attivo da giugno 2015.

Il secondo sistema, invece, permetteva la navigazione all'interno della classificazione Ateco 2007 ed era abbinato ad un tool di ricerca testuale. Questo applicativo era completamente integrato nel codice del sito e faceva uso di un sistema di ricerca all'interno di un database che riportava l'intera classificazione Ateco.

Questa applicazione è stata completamente riscritta utilizzando la classificazione Ateco 2007 in formato xml in modo da rendere la ricerca testuale una funzionalità integrata nel nuovo front-end.

Inoltre è stata migliorata la navigazione gerarchica che offre agli utenti la possibilità di visualizzare l'intera classificazione Ateco attraverso una struttura ad *accordion.*[19]

Infine è stata inserita una nuova funzionalità, molto importante per gli utenti, ed ossia la possibilità di effettuare la ricerca dell'attività economica a partire dal codice Ateco.

6.2 Lo sviluppo

Per realizzare il sistema di codifica delle attività economiche sono state sviluppate sia la parte server (back-end) sia quella client (front-end), che dialogano tramite web service. L'utilizzo di questo sistema di interscambio dati presenta il vantaggio di rendere la parte pubblica (front-end) completamente svincolata dai dati (server), così da consentire qualunque modifica nella visualizzazione, senza dover intervenire sul sistema di accesso ai dati.

L'applicazione è stata scritta facendo largo uso di javascript e di j*query* per fornire in maniera asincrona tutte le tipologie di risposte cercate dall'utente, in modo da non distogliere la sua attenzione dalla pagina centrale nella quale lavora l'applicazione e da alleggerire notevolmente il lavoro del server.

Il front-end è stato sviluppato utilizzando come fonte dati il web service ACTR che fornisce il risultato già formattato in html. Per semplificare il successivo passaggio all'utilizzo del web service WITCH-CIRCE, tale output è stato destrutturato dalla sua componente di markup. Il web service WITCH-CIRCE fornisce infatti le risposte in formato json utilizzabile direttamente all'interno di applicazioni javascript. Separando la logica sottesa all'elaborazione dei dati dalla formattazione degli stessi è stato possibile emulare il comportamento di WITCH-CIRCE rendendo così il front-end compatibile con entrambe le fonti dati. In questa maniera la visualizzazione del risultato della classificazione viene demandato interamente alla combinazione di html e CSS presente all'interno del front-end.

6.3 Il funzionamento

Nei paragrafi successivi vengono descritte tutte le funzionalità offerte dal nuovo front end:

1. l'individuazione del codice di attività economica;
2. la ricerca per codice di attività;
3. la ricerca testuale;
4. l'area di download.

6.3.1 Individuazione del codice di attività economica

Per la codifica dell'attività economica l'utente interroga il front-end inserendo la stringa da lui cercata nell'apposito campo, evidenziato in verde nella figura 28.

[19] Accordion rappresenta un meccanismo a fisarmonica che all'apertura di una voce chiude quella precedentemente aperta.

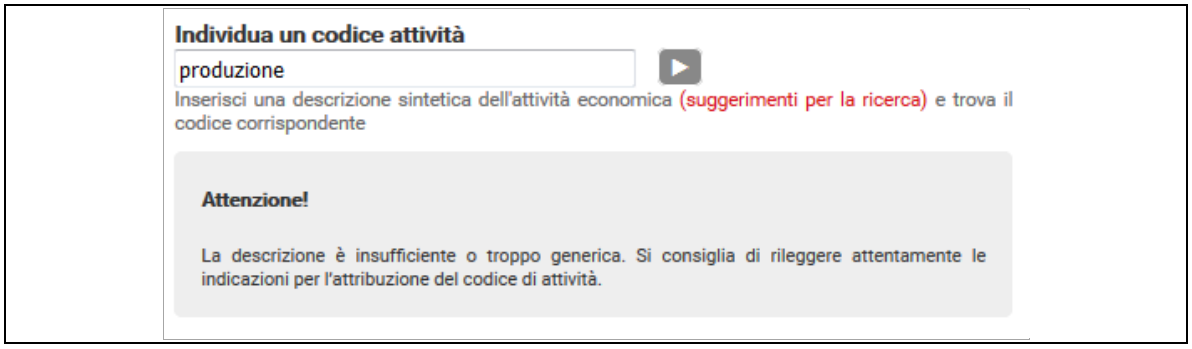
Figura 28 - Funzione di codifica della variabile Ateco

Dopo opportune verifiche, la stringa digitata dall'utente viene inviata al web service il quale risponde con un contenuto differente in base al testo ricercato:

1. un codice di errore qualora la stringa sia troppo generica o non presente all'interno del sistema di classificazione;
2. una o più definizioni affini alla stringa ricercata tra le quali scegliere.

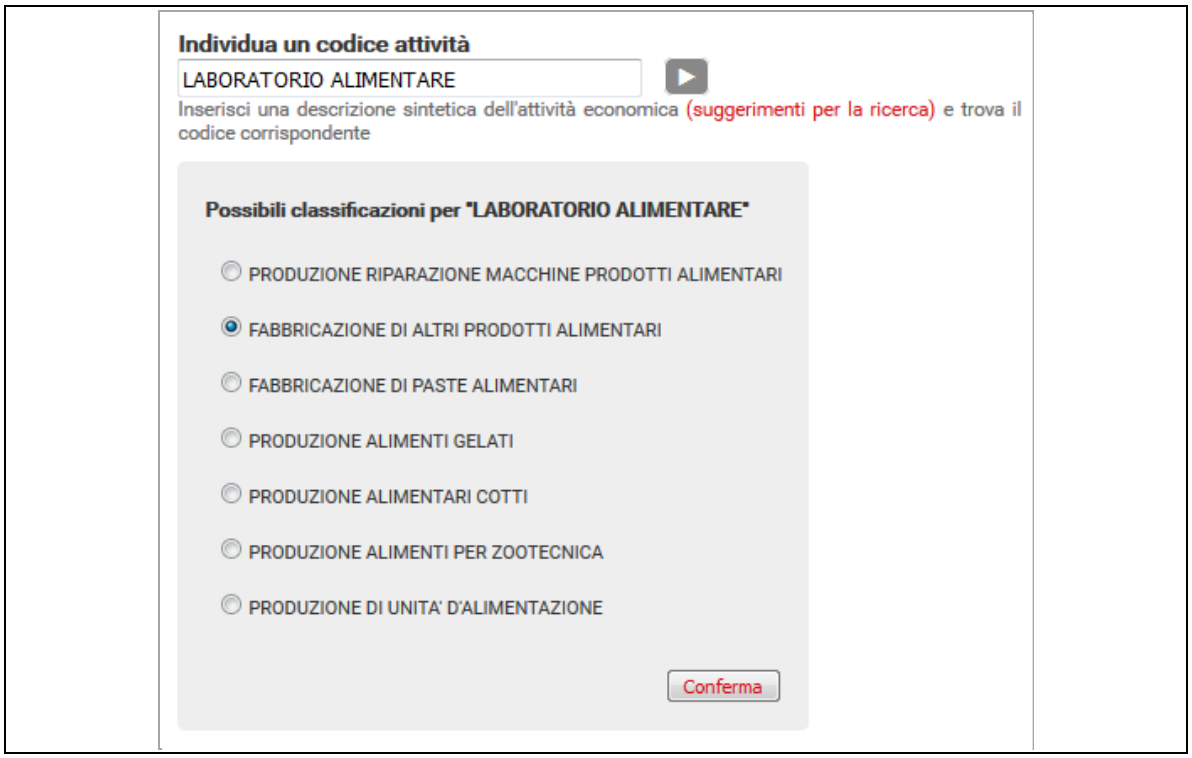
Nel primo caso il codice prodotto dal web service viene interpretato dal front-end e presentato all'utente sotto forma di messaggio (Figura 29).

Figura 29 - Esempio di messaggio di errore che si presenta all'utente



Nel secondo caso viene visualizzata la descrizione di tutte le voci che il sistema WITCH-CIRCE associa alla definizione selezionata (figura 30).

Figura 30 - Risultato della codifica WITCH-CIRCE



È compito del front-end collegare il codice Ateco a sei cifre restituito da WITCH-CIRCE alla descrizione testuale. Tale collegamento avviene attraverso il contenuto di un file, anch'esso in formato json. I dati all'interno di questo file sono gli stessi utilizzati dalla precedente applicazione, solo che mentre prima erano memorizzati in un DataBase MySql, in questo caso si è scelto di archiviarli su file dal momento che l'utilizzo è in sola lettura e le attività di manutenzione, quali inserimento e modifica dati, non rientrano nei requisiti del progetto. Del resto, l'eliminazione del DataBase MySql per la decodifica dei codici Ateco, oltre ai vantaggi architetturali appena esposti, porta all'eliminazione di potenziali rischi di sicurezza.

Le descrizioni così ottenute vengono visualizzate dall'utente nell'ultima schermata del processo di identificazione del codice attività (figure 30 e 31).

Figura 31 - Codice Ateco completo visualizzato dopo la conferma

Per evitare di rendere la pagina troppo densa di informazioni e facilitare quindi la lettura dei risultati, vengono visualizzate solo una parte delle informazioni necessarie a capire se è stato individuato il codice pertinente con l'attività economica digitata. Attraverso il comando "Continua…" presente nell'interfaccia (Figura 31) è possibile leggere la restante parte delle informazioni di dettaglio.

In quest'ultimo passaggio è stato inserito il comando "< indietro" per tornare alla schermata precedente di selezione della stringa di classificazione. Utilizzando questa funzionalità si mantiene comunque traccia dell'ultima selezione effettuata così da facilitare il lavoro di ricerca e selezione del codice Ateco corrispondete all'attività economica inserita.

6.3.2 Ricerca per codice attività e visualizzazione gerarchica

Entrambe le tipologie di ricerca si basano sullo stesso file in formato xml che contiene l'intera classificazione Ateco in forma gerarchica.

La visualizzazione gerarchica mostra sul pc dell'utente l'intero albero, mentre la ricerca per codice mostra soltanto l'albero di derivazione relativo alla voce cercata.

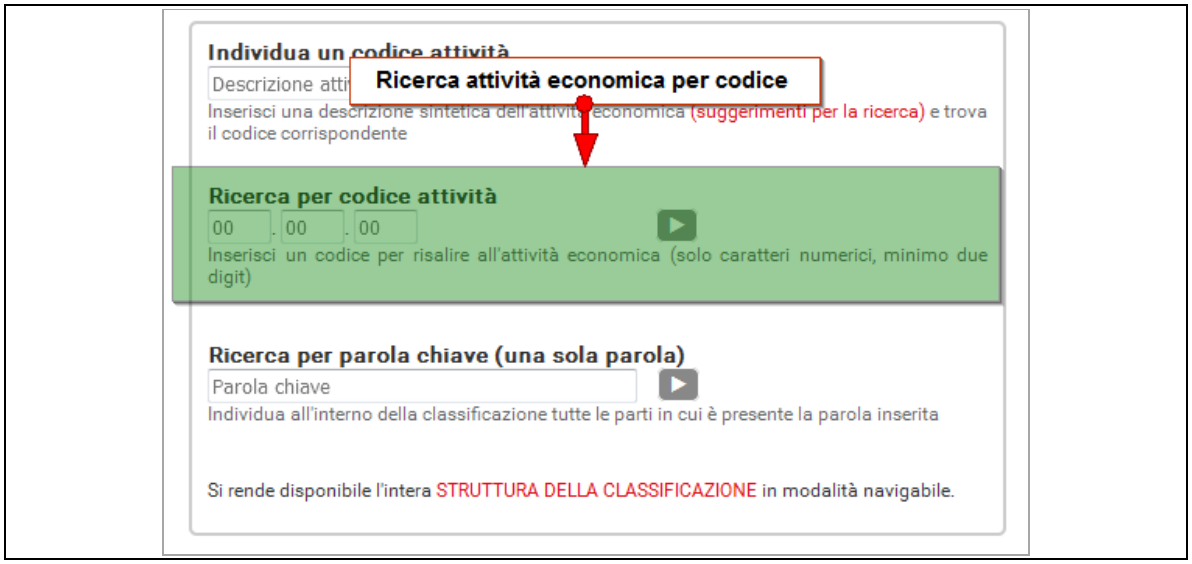
Attraverso l'utilizzo delle funzioni di x*query* in php viene trasformato il formato xml iniziale in un formato html. Tale formato è stato strutturato in modo da poter essere utilizzato dal sistema di accordion fornito da j*query*-UI.

Figura 32 - Visualizzazione dell'intera struttura classificazione

La visualizzazione dell'intera classificazione è disponibile in un'unica schermata compatta, che si espande cliccando su ciascuna voce. Utilizzando l'icona disponibile alla destra del testo è inoltre possibile leggere la descrizione di ogni voce (figura 32).

La ricerca per codice testuale sfoglia tutti i nodi del documento xml alla ricerca del codice a due, quattro o sei cifre inserito dall'utente (figura 33).

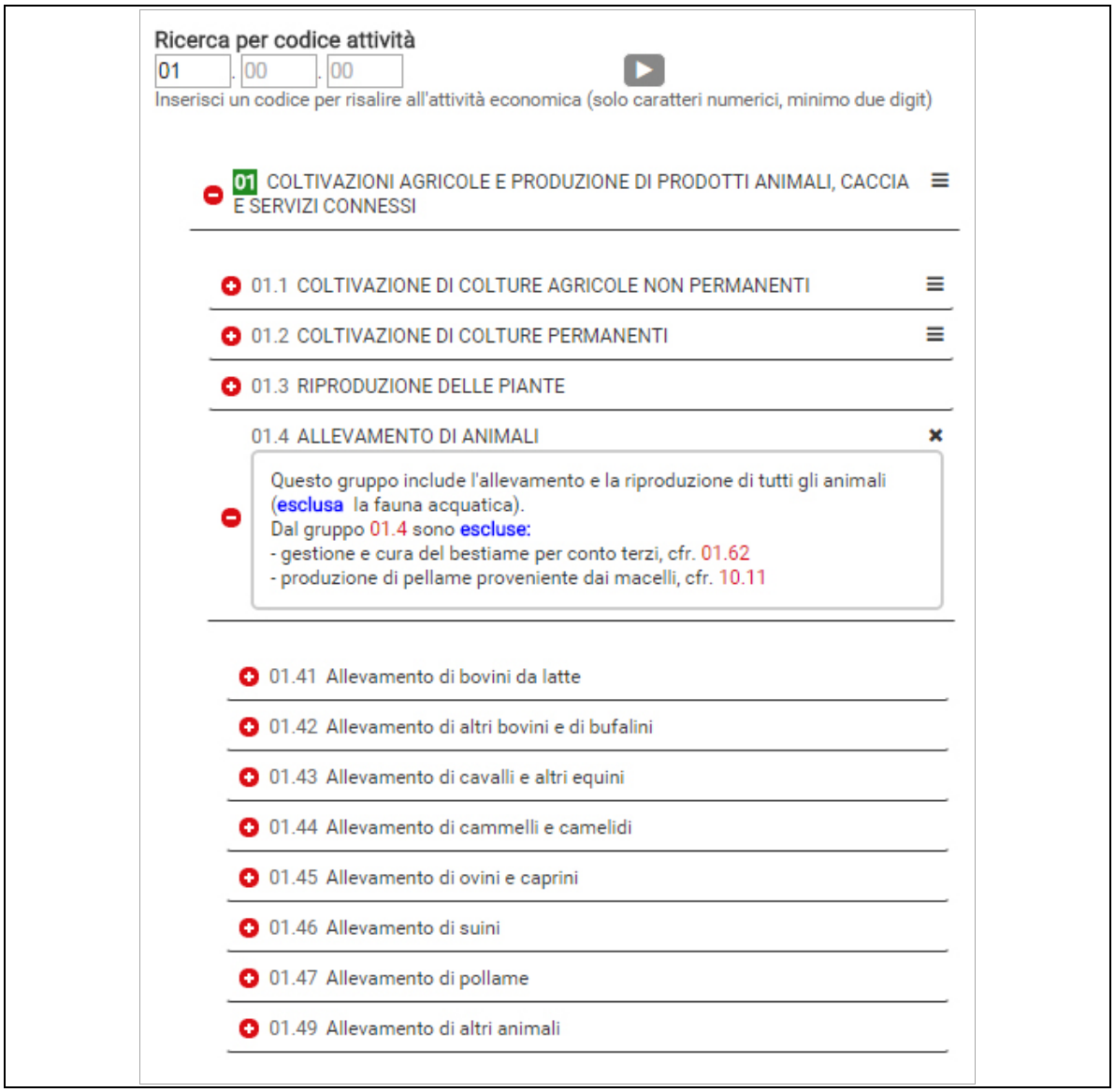
Figura 33 - Ricerca per codice di attività



Il front-end, in fase di visualizzazione, provvede a effettuare alcune trasformazioni sul testo, rendendo più facilmente riconoscibili le esclusioni dalla descrizione e fornendo un collegamento ipertestuale ogni volta che all'interno di una descrizione viene inserito un riferimento ad un altro codice Ateco.

Nell'esempio, in figura 33, è presente una ricerca effettuata sul codice 01, con la visualizzazione della sola voce relativa a questo codice e di una descrizione aperta dove sono, evidenziate in blu le voci escluse e in rosso i codici Ateco che fungono da collegamento ipertestuale.

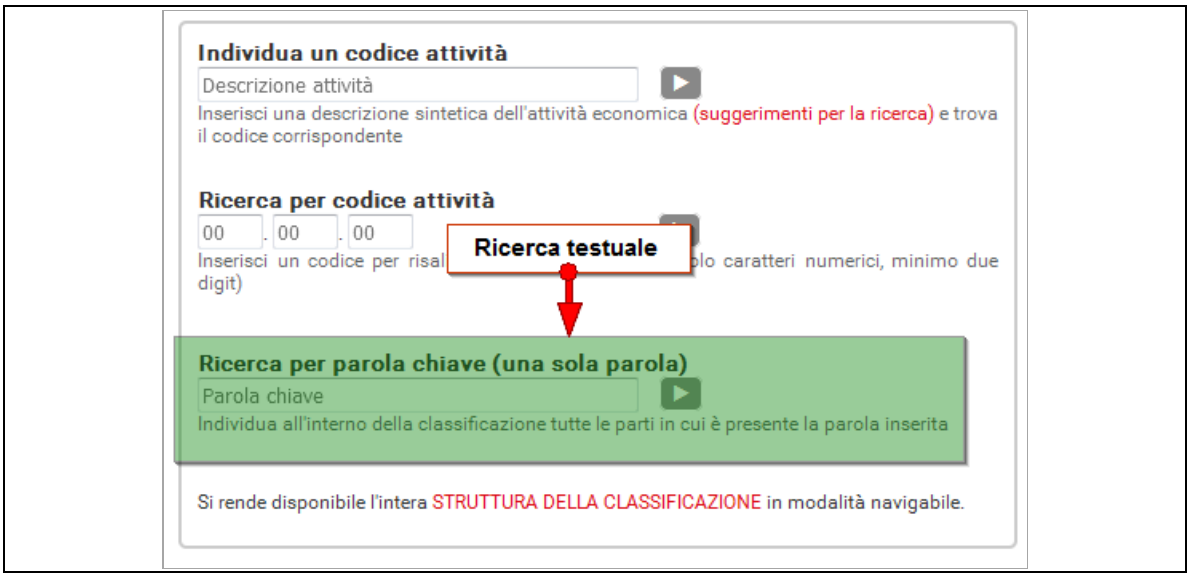
Figura 34 - Una visualizzazione tipo della ricerca per codice attività



6.3.3 Ricerca testuale o per parola chiave

A partire dalla stessa base dati in formato xml è stato creato anche il nuovo strumento di ricerca testuale (Figura 35). Il procedimento tramite il quale il sistema associa la stringa cercata dall'utente alle occorrenze della stessa è una combinazione del già citato x*query* con un sistema di espressioni regolari.

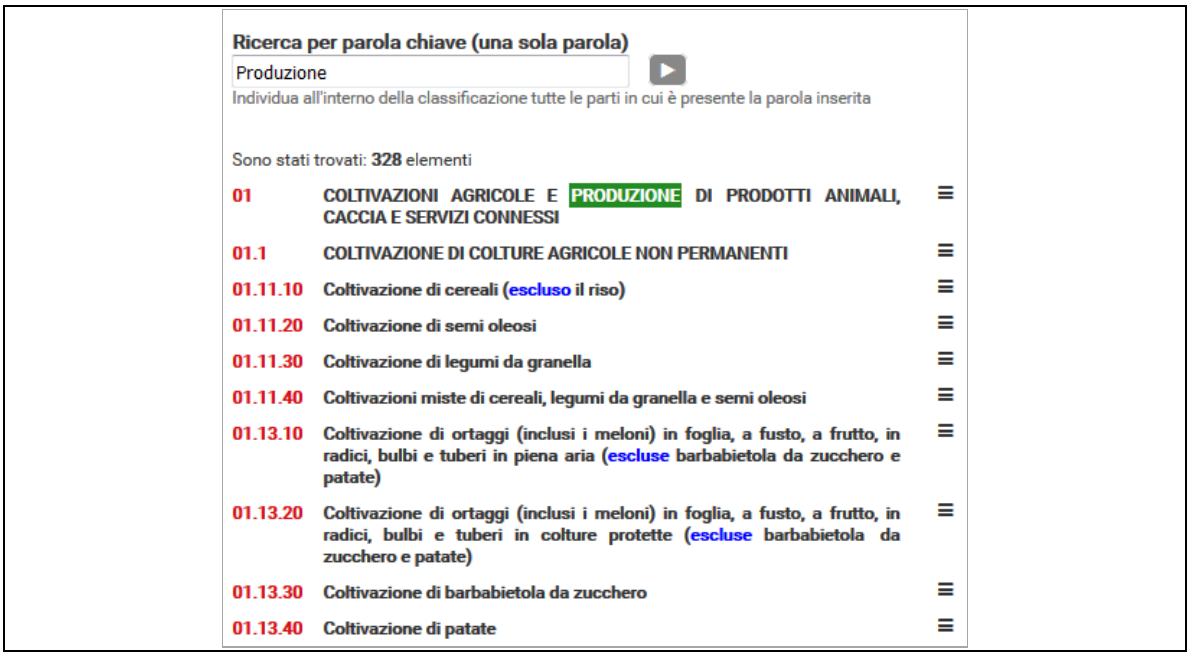
Figura 35 - Ricerca testuale o per parola chiave



I risultati di questa ricerca vengono poi mostrati in modo da evidenziare sia le occorrenze della stringa cercata sia i prodotti o le attività escluse dalle categorie e/o sottocategorie dell'Ateco (Figura 36).

Come per l'individuazione del codice attività tramite interazione con WITCH-CIRCE, anche in questo caso, eventuali link ad altri voci dell'Ateco si attivano sotto forma di collegamento ipertestuale.

Figura 36 - Occorrenze della voce cercata (in verde) ed esclusioni



6.3.4 *Area download*

In fondo alle schermate dell'applicazione è presente un box contenente tutta la documentazione relativa alla classificazione Ateco (Figura 37).

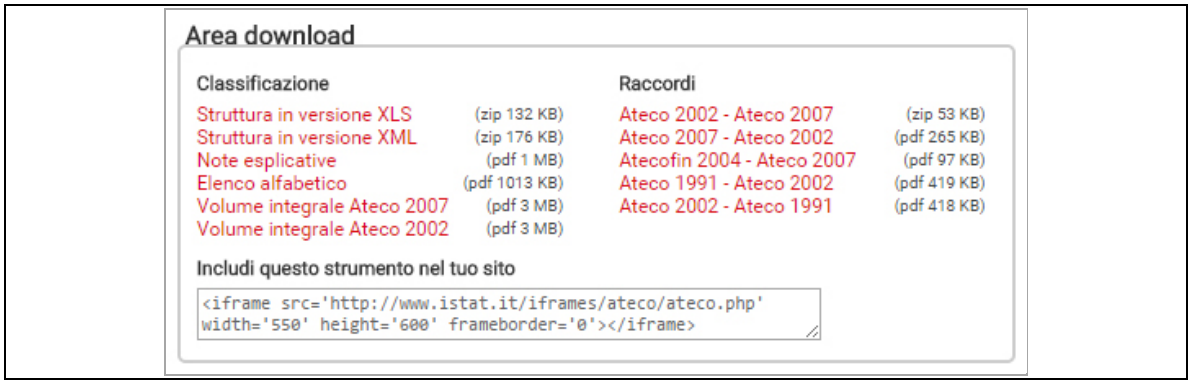
Si tratta di una lista statica di documenti a corredo del front-end che viene visualizzata anche dagli utenti che incorporano l'applicazione all'interno del loro siti, così da garantire, su qualsiasi

piattaforma, la raggiungibilità di tutte le informazioni utili.

Sono presenti, in particolare, la struttura della classificazione in formato xls e xml, le note esplicative della classificazione, l'elenco alfabetico delle voci comprese nelle sottocategorie di attività economiche e i raccordi tra diverse versioni di Ateco.

In fondo al box è offerta la possibilità di inserire all'interno del proprio sito web, tramite iframe, tutto il front-end.

Figura 37 - Contenuto dell'area download



6.4 L'accessibilità

L'obiettivo principale della nuova applicazione è stato quello di soddisfare i requisiti di accessibilità previsti dalle WCAG 2.0 (level AAA), le Web Content Accessibility Guidelines (http://www.w3.org/TR/WCAG) redatte dal W3C nel 2008. Si tratta di un'ampia gamma di raccomandazioni studiate per rendere i contenuti web accessibili alle persone con disabilità, tra cui cecità e ipovisione, sordità, limitazioni cognitive e dell'apprendimento, ridotte capacità di movimento, disabilità della parola, fotosensibilità.

Si è lavorato dunque per fare in modo che:

1. i campi in cui inserire i testi o i codici fossero selezionabili sia con il mouse sia tramite tastiera (attraverso l'uso del tasto tab);
2. l'intera classificazione fosse navigabile anche da tastiera (attraverso l'uso dei tasti tab + invio);
3. ReadSpeaker, il sistema di lettura dei testi disponibile sul sito istituzionale, fosse in grado di leggere anche il contenuto caricato nella pagina attraverso javascript, cioè anche i risultati che si modificano sulla base dell'interazione dell'utente.

Per raggiungere il primo obiettivo si è lavorato sul form, strutturandolo in modo completo secondo la struttura relativa alla documentazione W3C in merito ai documenti in html5, si è inoltre provveduto alla personalizzazione di alcune funzioni javascript che consentono, utilizzando la sola tastiera, di procedere all'invio del form compilato e alla navigazione tra gli elementi, con particolare attenzione ai campi della ricerca per codice, altrimenti di difficile utilizzo.

Per ottenere il secondo risultato ci si è invece affidati alle funzionalità offerte dal framework j*query*-UI che è strutturato in modo da offrire, sulla maggior parte dei browser moderni, le funzionalità di indicizzazione dei tab in modo da poter garantire la navigabilità da tastiera dei contenuti rappresentati all'interno dell'*accordion*. Questa soluzione tecnica è già largamente utilizzata all'interno del sito istituzionale.

Il raggiungimento dei primi due obiettivi, oltre a garantire l'accessibilità del nuovo front-end, è stato anche un grande passo avanti nell'usabilità dello stesso.

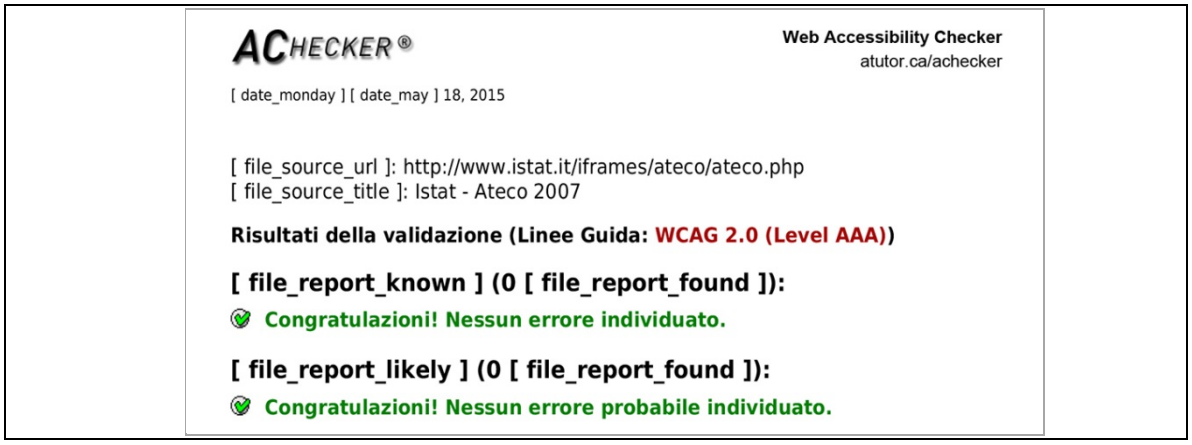
Per rendere disponibile il sistema di lettura dei contenuti del sito, l'intero risultato visualizzato dall'utente al termine dell'esecuzione degli script viene selezionato e inviato a ReadSpeaker che (soltanto in quel momento) procede con la lettura. Questo accorgimento consente di rendere disponibile i contenuti dell'applicazione anche ad ipovedenti, nonostante l'utilizzo intensivo di javascript.

Per verificare la conformità del contenuto web alle direttive di accessibilità si è ricorsi al Web

Accessibility Checker (http://achecker.ca/checker/index.php), uno degli strumenti suggeriti dallo stesso W3C in grado di segnalare sia gli errori noti con certezza come barriere all'accessibilità, sia gli errori probabili che richiedono però l'intervento umano per decidere se sia effettivamente necessario modificare la pagina html.

La nuova applicazione soddisfa pienamente i requisiti di accessibilità (level AAA) previsti dalle WCAG 2.0 (Figura 38)

Figura 38 – Conformità ai requisiti di accessibilità



Per quanto riguarda le linee guida della legge Stanca, invece, l'unico problema, di natura più legale che tecnica, che resta da risolvere è legato all'intestazione di documenti: html5.

Tali documenti per loro natura non prevedono una versione "strict" richiesta invece dalla legge stessa, la quale però suggerisce anche di adottare i più moderni accorgimenti tecnici per la creazione di documenti da pubblicare nei siti web della pubblica amministrazione.

L'applicazione è stata comunque strutturata in modo tale da rendere piuttosto semplice il ritorno all'xhtml-strict qualora si dovesse stabilire che la volontà del legislatore sia effettivamente questa.

7 Sviluppi futuri

In questo capitolo si è voluto dedicare spazio a potenziali linee di ricerca che potrebbero nascere sulla base dell'esperienza maturata. In particolare, nelle pagine seguenti, vengono delineate due tipologie di progetti di ricerca e sviluppo che riguardano, rispettivamente, possibili integrazioni dell'algoritmo di codifica alla base di CIRCE (paragrafo 7.1 e seguenti) e possibili implementazioni di web service che offrano servizi di codifica diversificati per variabili e/o per modalità di navigazione e codifica delle stesse (paragrafo 7.2 e seguenti).

7.1 L'ottimizzazione del sistema di codifica

L'ottimizzazione del pacchetto di codifica CIRCE, descritta nei paragrafi successivi riguarda, da un lato, il miglioramento dell'efficienza del sistema in termini di tempi di codifica e, dall'altro, lo studio di altri algoritmi di *matching* che possano integrare o sostituire quello attuale.

Il problema dei tempi di codifica è particolarmente sentito per la modalità *batch* e per la fase di monitoraggio della qualità. Per cercare di risolverlo sono allo studio diverse soluzioni come la parallelizzazione di alcune funzioni di CIRCE o la loro implementazione con linguaggi compilati.

L'algoritmo di *matching* di CIRCE è uno dei possibili algoritmi utilizzabili per il confronto fra stringhe. L'algoritmo ricalca quello di ACTR v3. Tale scelta, effettuata per mantenere continuità di servizio agli utenti del sistema di codifica, non impedisce che possano essere utilizzati altri algoritmi di *matching* al fine di migliorare alcune caratteristiche del sistema, ferme restando l'efficacia e la qualità raggiunte. Un tipo di soluzione allo studio, descritta nel seguito di questo capitolo, è l'utilizzo di funzioni di similarità fra stringhe, per le quali è possibile sfruttare l'esperienza maturata nel campo del *record linkage*.

7.1.1 L'ottimizzazione dei tempi della codifica batch

Come accennato nel paragrafo 2, il nuovo sistema di codifica CIRCE è stato progettato e realizzato allo scopo di sostituire il software ACTR v3 nei processi di produzione dell'Istat. Da un punto di vista tecnologico, la migrazione da ACTR v3 a CIRCE ha comportato la transizione da un'applicazione sviluppata in un linguaggio compilato, il C, ad una sviluppata in un linguaggio interpretato, R.

Come è noto, la natura di linguaggio interpretato di R penalizza in modo particolare l'efficienza di programmi che facciano ricorso esplicito ad istruzioni di ciclo che risultano almeno 2 ordini di grandezza più lenti degli omologhi sviluppati in C. D'altra parte, R dispone di metodi di vettorizzazione molto efficaci e flessibili, che consentono, nella maggior parte dei casi, di evitare le strutture di ciclo, garantendo significativi incrementi di performance.

In virtù di queste considerazioni, le funzioni di confronto e *scoring* di CIRCE (vedi paragrafo 2.5), che costituiscono le componenti di maggiore complessità computazionale del sistema, sono state integralmente programmate in codice R vettorizzato. Rispetto al codice R prototipale che faceva uso di cicli, la vettorizzazione del codice ha determinato un incremento di efficienza di un fattore 15 nel "caso medio" (cioè per stringhe di input di media complessità) e di un fattore 250 nel "caso peggiore" (cioè per stringhe di input di elevata complessità).

Lo *speedup*[20] così conseguito è stato cruciale per l'effettiva messa in esercizio della nuova applicazione web per la codifica on-line dell'Ateco. Se, infatti, il tempo medio richiesto da CIRCE per la codifica di una singola stringa di input resta, in termini relativi, significativamente superiore a quello di ACTR v3, la differenza assoluta fra i tempi di risposta dei due sistemi è ormai inavvertibile a livello utente (centesimi di secondo anziché millesimi di secondo). Inoltre, l'efficienza raggiunta dall'attuale implementazione di CIRCE è tale da garantire che le prestazioni dell'applicazione web rimangano adeguate anche in condizioni di picco del carico (numero molto elevato di connessioni http concorrenti). Nel complesso, dunque, l'impatto della transizione da ACTR v3 a CIRCE sulla performance della codifica on-line dell'Ateco può essere considerato trascurabile.

Al contrario, l'efficienza di CIRCE non è ancora pienamente soddisfacente nelle procedure di codifica *batch* di grandi file (decine di migliaia di stringhe di input), generalmente eseguite in Istat in ambiente pc. In questo contesto applicativo, la differenza nei tempi di risposta di CIRCE ed ACTR v3 è tuttora significativa: decine di minuti contro minuti. Tale differenza di performance è integralmente ascrivibile alla diversa implementazione dei rispettivi linguaggi di programmazione (R interpretato, C compilato) e non può essere ridotta ottimizzando ulteriormente il codice R di CIRCE. È, tuttavia, possibile immaginare almeno due diverse alternative per accelerare i tempi di risposta :

1. la parallelizzazione delle funzioni di confronto e *scoring*;
2. l'implementazione delle funzioni di confronto e *scoring* in codice compilato (Fortran o C).

Ciascuna delle alternative presenta vantaggi e costi specifici. Ad esempio, il ricorso al codice compilato garantisce incrementi di efficienza apprezzabili anche in ambienti PC con processori *single-core*, laddove i benefici della parallelizzazione divengono effettivi solo su macchine *multi-core* o su architetture *cluster*. Al contrario, mentre è possibile parallelizzare il codice R interpretato in modo indipendente dalla piattaforma di calcolo su cui il codice è eseguito, le chiamate a codice compilato richiedono la costruzione e la manutenzione di librerie dinamiche dipendenti dalla macchina e dal sistema operativo (file .dll in Windows e .so in Linux e nei sistemi Unix-like). In ultimo, lo sforzo di programmazione necessario per re-implementare le funzioni *time-critical* di CIRCE in Fortran o C eccede di gran lunga quello richiesto per parallelizzarne il codice R. Nel complesso, il ricorso al codice compilato appare l'alternativa sfavorita in virtù dei maggiori costi di programmazione e manutenzione. A favore della parallelizzazione gioca anche il fatto che la totali-

20 Nel presente contesto, per 'speedup' si intende l'incremento relativo di velocità, nell'esecuzione di un task fissato, determinato da uno specifico intervento di ottimizzazione (ad esempio, la vettorizzazione del codice di un programma R).

tà dei pc dell'Istat sia ormai dotata di processori *multi-core*.

Fortunatamente, lo *"scoring engine"* di CIRCE (motore di calcolo che include le funzioni di confronto fra stringhe, calcolo dei punteggi e *ranking*) implementa algoritmi in larga misura parallelizzabili. In aggiunta, la loro conversione da codice sequenziale a codice parallelo non presenta difficoltà tecniche di particolare rilievo. Quindi è relativamente semplice decomporne i singoli task di calcolo in subtask indipendenti la cui esecuzione può essere demandata a nodi di calcolo distinti che operano simultaneamente. L'incremento di performance, idealmente lineare nel numero di nodi, sarà in genere sub-lineare a norma della legge di Amdahl.

Prima di optare definitivamente per l'ingegnerizzazione (e la successiva messa in esercizio) di una versione parallela dello *scoring engine* di CIRCE, è stato ritenuto opportuno eseguire un *Prof of Concept* (PoC) volto a verificarne la fattibilità tecnica e quantificarne i benefici. Il PoC si è avvalso dell'infrastruttura standard di calcolo parallelo disponibile in R, il package *'parallel'* (R Core Team 2015), integrato nella distribuzione base del linguaggio sin dal 2011 (in occasione della pubblicazione di R 2.14.0).

Le caratteristiche sperimentali del PoC possono essere sinteticamente descritte come segue:

- Task del PoC: Codifica Ateco *batch*
- Voci del dizionario: 34.073
- Stringhe di input: 36.919
- Architettura di elaborazione: PC (CPU 64-bit, 2 core, clock 3.1 GHz, RAM 8 GB, SO Windows 7 64-bit)
- Build di R: 64-bit

I risultati ottenuti nel PoC sono riportati nella tavola 21.

Tavola 21 - Tempi di esecuzioni dei moduli di CIRCE e incrementi di performance indotti dalla parallellizzazione dello Scoring Engine

Modulo di CIRCE	Tempo di Esecuzione (minuti)		Incremento di performance (%)	
	Codice Sequenziale	Codice Parallelo	$\Delta T / T = (T_{par} - T_{seq}) / T_{seq}$	Speedup = $T_{seq} / T_{par} - 1$
Parsing	3	3	0,00%	0,00%
Scoring Engine	35	22	- 37,00%	**59,00%**
Scrittura Output	12	12	0,00%	0,00%
Totale	50	37	- 26,00%	**35,00%**

Sebbene l'unico modulo di CIRCE sottoposto a conversione da codice sequenziale a codice parallelo sia stato lo *scoring engine*, la tavola 21 mostra anche i tempi di esecuzione dei restanti moduli. È, così, possibile apprezzare sia l'incremento di efficienza indotto dalla parallelizzazione sullo *scoring engine* stesso, sia il conseguente aumento complessivo di performance per CIRCE: i relativi speedup percentuali sono riportati in grassetto. Su una macchina con due soli core, l'accelerazione dello *scoring engine* è certamente significativa (speedup del 59% e tempi di risposta ridotti del 37%), così come apprezzabile è l'impatto sulla velocità di CIRCE nel suo complesso (speedup del 35% e tempi di risposta ridotti del 26%). I risultati dello studio di fattibilità sono, dunque, positivi ed incoraggiano ad investire nell'ingegnerizzazione e nella messa in esercizio del codice parallelo prototipale sviluppato per il PoC.

Concludiamo questo paragrafo offrendo un'analisi essenziale del livello di *scalabilità* che CIRCE potrebbe acquisire a seguito della parallelizzazione dello *scoring engine*. È, infatti, evidente che l'installazione dell'applicazione su architetture di calcolo più potenti di quelle utilizzate per il PoC (PC con più di 2 core o server di produzione con più di 2 CPU) debba garantire il conseguimento di ulteriori aumenti di efficienza.

La legge di Amdahl modella lo speedup teorico indotto dalla parallelizzazione, S, come funzione del numero di nodi di calcolo dell'architettura su cui l'applicazione è eseguita, N, e della frazione di calcolo effettivamente parallelizzata, F:

$$S = S(F, N) = \frac{1}{1 - F + (F/N)} - 1 \tag{5}$$

Questa relazione formalizza l'osservazione intuitiva che esiste un limite superiore allo speedup ottenibile parallelizzando un'applicazione:

$$S(F, \infty) = \frac{F}{1 - F} \tag{6}$$

Il limite (6) deriva direttamente dal fatto che i benefici della parallelizzazione sono circoscritti ad una certa frazione, $F \leq 1$, del calcolo eseguito. Ora, noto lo speedup misurato empiricamente su una certa architettura, la legge di Amdahl (5) consente di ricavare una stima della frazione di calcolo effettivamente parallelizzata:

$$F = F(S, N) = \frac{S}{1 + S} \cdot \frac{N - 1}{N} \tag{7}$$

Sostituendo nella formula precedente lo speedup osservato per lo *scoring engine* ($S = 0.59$) nel PoC – cioè su una macchina con $N = 2$ core – è dunque possibile stimarne la frazione effettivamente parallelizzata in:

$$F_{SE} = F(0.59, 2) \cong 0.74 \tag{8}$$

Allo stesso modo, in base allo speedup osservato per CIRCE nel suo complesso ($S = 0.35$), è possibile dedurre che solo metà (circa) delle operazioni eseguite da CIRCE nella codifica *batch* è stata effettivamente parallelizzata nel PoC:

$$F_{CIRCE} = F(0.35, 2) \cong 0.52 \tag{9}$$

Ora, come riportato nella tavola 21, lo speedup misurato su una certa architettura, $S = T_{seq} / T_{par} - 1$, e la corrispondente riduzione relativa del tempo di esecuzione, $\Delta T / T = (T_{par} - T_{seq}) / T_{seq}$, sono legati dalla relazione:

$$\frac{\Delta T}{T} = -\frac{S}{1 + S} \tag{10}$$

Ne consegue che, sostituendo le stime di F_{SE} ed F_{CIRCE} nella legge di Amdahl (5), è possibile predire gli speedup attesi per lo *scoring engine* e per CIRCE su architetture dotate di N nodi di calcolo, nonché, in virtù della relazione precedente (6), le corrispondenti diminuzioni percentuali dei

tempi di risposta. Una selezione di tali previsioni è riportata nella tavola 22.

Si noti che, in virtù dell'equazione (10), il massimo speedup teoricamente ottenibile per CIRCE è solo di poco superiore al 100%:

$$S(F_{CIRCE}, \infty) \cong 1.08 \tag{11}$$

Tavola 22 - Previsione degli incrementi di performance ottenibili, nella codifica *batch* dell'Ateco, eseguendo CIRCE su architetture parallele con N nodi di calcolo.

N	Scoring Engine		CIRCE	
	Speedup (%)	ΔT / T (%)	Speedup (%)	ΔT / T (%)
1	0,00%	0,00%	0,00%	0,00%
2	59,00%	- 37,00%	35,00%	- 26,00%
4	125,00%	- 56,00%	64,00%	- 39,00%
8	184,00%	- 65,00%	83,00%	- 46,00%
10	199,00%	- 67,00%	88,00%	- 47,00%
12	211,00%	- 68,00%	91,00%	- 48,00%
14	220,00%	- 69,00%	93,00%	- 48,00%
16	227,00%	- 69,00%	95,00%	- 49,00%
32	253,00%	- 72,00%	102,00%	- 50,00%

Come si vede, se CIRCE venisse installato su un'architettura con 8 nodi di calcolo, i tempi di risposta della codifica *batch* risulterebbero pressoché dimezzati rispetto all'implementazione sequenziale. Tuttavia, gli ulteriori incrementi di performance conseguibili con la transizione a macchine più potenti (dotate di 10 o più nodi di calcolo) sarebbero, tutto sommato, marginali. In effetti, lo speedup asintotico sarebbe, a tutti gli effetti pratici, già raggiunto per $N \gtrsim 12$.

7.1.2 Possibili linee di sviluppo: potenzialità delle funzioni di somiglianza fra stringhe

Come descritto nel capitolo 2, l'algoritmo di *matching* di CIRCE ricalca sostanzialmente quello di ACTR v3, il quale, nel confrontare le stringhe di input da codificare con quelle contenute nel dizionario di riferimento della classificazione, utilizza la funzione di uguaglianza.

Prima della fase di confronto, tutti i testi – sia le stringhe da codificare, sia le voci del dizionario – vengono sottoposti alla fase di standardizzazione. Tale fase è finalizzata a rendere identiche, ai fini dell'attribuzione di un codice, due stringhe che, pur sintatticamente diverse, siano caratterizzate dallo stesso contenuto semantico.

Successivamente, le stringhe standardizzate vengono fornite in input all'algoritmo di *matching*, che confronta le parole che compongono ciascuna stringa da codificare con tutte le parole che compaiono nelle voci del dizionario. A parità di lunghezza delle stringhe e di importanza o peso delle parole componenti, l'algoritmo di *matching* accoppierà ciascuna stringa da codificare con le voci del dizionario che contengono il maggior numero di parole uguali a quelle della stringa di input.

Gli algoritmi di codifica che adottano il confronto per uguaglianza presentano, al contempo, vantaggi ed elementi di debolezza. Lo svantaggio principale è, probabilmente, l'impossibilità di gestire adeguatamente gli errori ortografici in via automatica, a differenza di quanto avverrebbe in un sistema di codifica manuale. Infatti, per un operatore umano, il significato di una parola resta solitamente comprensibile, anche qualora la parola sia affetta da errori ortografici, come l'omissione di una lettera o l'inversione di due lettere. Al contrario, il confronto automatico per uguaglianza rende ogni parola affetta da errori ortografici irrimediabilmente diversa dalla sua versione originale esatta, alterando in modo potenzialmente catastrofico l'esito dell'algoritmo di *matching*.

Proprio in virtù della vulnerabilità dell'algoritmo di *matching* di CIRCE (come, del resto, di ACTR v3) rispetto agli errori ortografici, l'Istat ha provveduto, nel tempo, a:

(i) costituire, arricchire e manutenere, per ciascuna classificazione, un "inventario" degli errori ortografici più frequenti commessi nelle procedure di codifica automatica;

(ii) sviluppare "contromisure" volte a correggere tali errori durante la fase di standardizzazione dei testi, cioè prima dell'esecuzione dell'algoritmo di *matching*.

A titolo di esempio: dall'analisi dei file di output prodotti nel tempo dalla procedura di codifica dell'Ateco, è stato possibile evincere come un numero consistente di errori di codifica (o di mancate codifiche) derivasse dall'occorrenza nella stringa di input della parola "prodruzione". È stato, pertanto, deciso di introdurre nel file dei sinonimi (utilizzato durante la fase di standardizzazione dei testi) la trasformazione diretta: "prodruzione" = "produzione".

Vale la pena osservare, incidentalmente, come entrambe le attività (i) e (ii) siano a tutt'oggi interamente non automatiche, cioè svolte da esperti operatori umani. D'altra parte, gli errori ortografici e/o di digitazione potrebbero anche essere corretti – almeno parzialmente – prima della fase di standardizzazione, sottoponendo i testi di input ad un programma automatico di controllo ortografico. L'adozione di questo approccio richiederebbe, tuttavia, di valutare con grande attenzione l'adeguatezza e l'accuratezza del correttore ortografico, anche in relazione al dominio semantico specifico della classificazione. In caso contrario, il rischio principale sarebbe l'eventuale creazione di "falsi positivi", ovvero *match* erronei derivanti da correzione ortografiche automatiche non adatte al contesto.

La limitazione dai "falsi positivi" costituisce, per altro, uno dei principali vantaggi e punti di forza del confronto per uguaglianza. Esso protegge, infatti, rispetto a possibili falsi *match* determinati da parole che, pur sintatticamente molto simili, abbiano, nel contesto della classificazione in oggetto, significato completamente diverso. Basti pensare, nel caso della classificazione Ateco, alle parole "banco" e "banca" che, nonostante differiscano nella sola vocale finale, afferiscono a settori diversi di attività economica, come mostrano le figure 39 e 40 seguenti.

Figura 39 - Codifica on-line parola "Banco" – Dizionario a 5 digit Ateco 2007

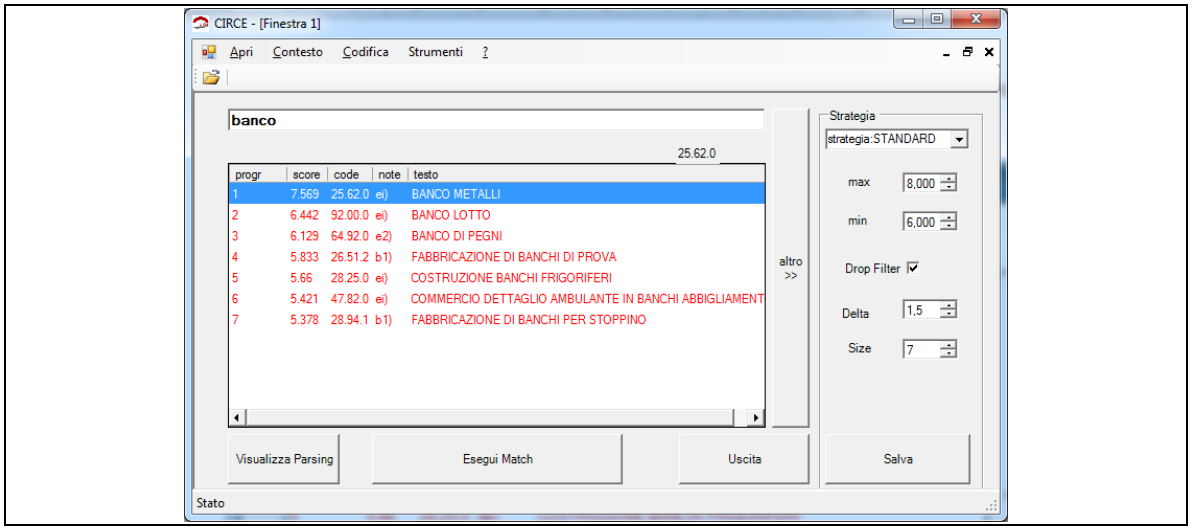
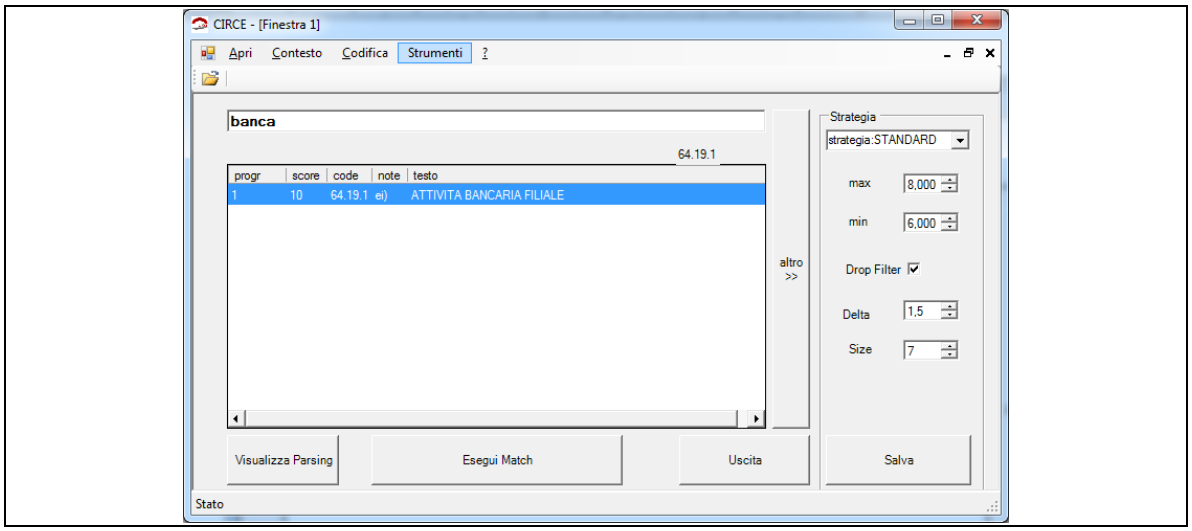


Figura 40 - Codifica on-line parola "Banca" – Dizionario a 5 digit Ateco 2007



Per alcune classificazioni, l'adozione nell'algoritmo di *matching* di funzioni di somiglianza fra stringhe potrebbe costituire una valida alternativa al confronto per uguaglianza.

Una funzione di somiglianza mappa coppie di stringhe t_1, t_2 in numeri reali $s_{12} = s(t_1, t_2)$, in modo che valori "bassi" di s_{12} corrispondano a coppie di stringhe marcatamente diverse, e viceversa. In genere, le funzioni di somiglianza vengono normalizzate in modo che il loro codominio, cioè l'insieme dei valori s_{12} possibili, coincida con l'intervallo chiuso e limitato [0, 1]. Inoltre, tranne rare eccezioni[21], la normalizzazione è tale che la somiglianza assuma valore massimo, $s_{12} = 1$, se e solo se $t_1 \equiv t_2$, cioè quando stringhe t_1 e t_2 coincidono.

Come è facile intuire, l'uso di funzioni di somiglianza potrebbe agevolare l'algoritmo di *matching* qualora nel testo da codificare e nelle voci di dizionario comparissero parole "simili" sia sintatticamente che semanticamente. Il beneficio sarebbe massimo per classificazioni concettualmente semplici (cioè non basate su criteri classificatori astratti), nei cui dizionari è raro riscontrare parole sintatticamente simili ma semanticamente diverse (la classificazione dei Comuni è un buon esempio). Sotto queste condizioni, il confronto per somiglianza avrebbe il pregio di attenuare, rispetto a quello per uguaglianza, il rischio di "falsi negativi" (cioè *match* mancati), senza tuttavia incrementare significativamente il rischio di "falsi positivi".

Ad esempio, nel caso della codifica del Comune, la versione attuale di CIRCE non riuscirebbe ad abbinare univocamente la stringa di input "San Felice Cirdeo" con la voce di dizionario "San Felice Circeo", a meno che nel file dei sinonimi non fosse esplicitamente prevista la trasformazione "Cirdeo" = "Circeo". Al contrario, il problema verrebbe certamente risolto – senza bisogno di trasformazioni preliminari – (i) dall'adozione di una opportuna funzione di somiglianza fra stringhe e (ii) dalla definizione di una adeguata soglia di accettazione, vale a dire un valore s_{th} tale che $s(t_1, t_2) > s_{th}$ implichi l'intercambiabilità fra le stringhe t_1 e t_2, ossia $t_1 = t_2$.

Dunque, come è possibile intuire dall'esempio precedente, l'uso di funzioni di somiglianza, oltre a contrastare i problemi di codifica indotti dagli errori ortografici, determinerebbe un rilevante effetto collaterale positivo, ovvero lo snellimento dei file dei sinonimi e delle operazioni necessarie alla loro manutenzione, operazioni che, vale la pena ripetere, sono interamente a carico degli utenti di CIRCE.

I vantaggi potenziali dell'introduzione in CIRCE delle funzioni di somiglianza fra stringhe devono, tuttavia, essere attentamente ponderati, in particolare per le classificazioni più complesse, come l'Ateco.

Nel caso della classificazione Ateco, i file di *parsing*, che guidano la fase di standardizzazione

[21] Un esempio è fornito dalle funzioni di somiglianza fonetiche (come quelle basate sugli algoritmi della famiglia soundex), che possono assumere valore massimo anche per coppie di stringhe sintatticamente diverse.

dei testi, contengono, oltre ai sinonimi, anche i cosiddetti "criteri classificatori". Un esempio di "criterio classificatorio" è fornito dalla regola che considera prevalente il settore produttivo rispetto al commercio (intuitivamente: se un bene non viene prodotto, non può essere venduto). È solo in virtù di tale regola che una stringa di input intrinsecamente ambigua, quale "Produzione e vendita di calzature", la quale menziona due diverse attività economiche, viene univocamente codificata da CIRCE nel settore Produzione. Da un punto di vista operativo, questa regola di prevalenza è codificata all'interno dei file di *parsing* in modo tale che tutte le attività che riportano la dicitura "Produzione e vendita prodotto x" vengono trasformate in "Produzione prodotto x".

Inoltre, le voci dei dizionari della classificazione Ateco contengono un numero non trascurabile di parole sintatticamente simili ma semanticamente molto diverse (si ricordi la coppia "banco" "banca"), che – non a caso – costituiscono "eccezioni" rispetto alle generali regole di standardizzazione, poiché non vengono sottoposte all'eliminazione di prefissi, suffissi e caratteri multipli. In questi casi, l'identificazione di una soglia di accettazione troppo bassa per la funzione di somiglianza adottata potrebbe determinare *match* erronei ("falsi positivi").

7.1.2.1 Un'ipotesi allo studio: le funzioni di somiglianza per il riconoscimento delle parole criterio

Come illustrato nel paragrafo precedente, l'impatto potenziale della sostituzione, nell'algoritmo di codifica di CIRCE, della funzione di uguaglianza con un'opportuna funzione di somiglianza fra stringhe, è non solo molto elevato, ma anche significativamente diverso a seconda della classificazione di riferimento. Si tratta di una scelta delicata, che potrà essere operata solo a valle dell'esecuzione di accurati ed estesi test di fattibilità empirici.

Per progettare adeguatamente i test di fattibilità, occorrerà, in via preliminare, stabilire:

a) su quali classificazioni concentrare la sperimentazione;
b) in quale "passo" del processo di codifica introdurre le funzioni di somiglianza;
c) quale sia la funzione di somiglianza più adatta alle peculiarità della classificazione selezionata;
d) quale soglia di accettazione adottare, onde "correggere" automaticamente le parole di input affette da errori ortografici.

I punti c) e d) sono tuttora allo studio, mentre per i punti a) e b) si è ritenuto di:

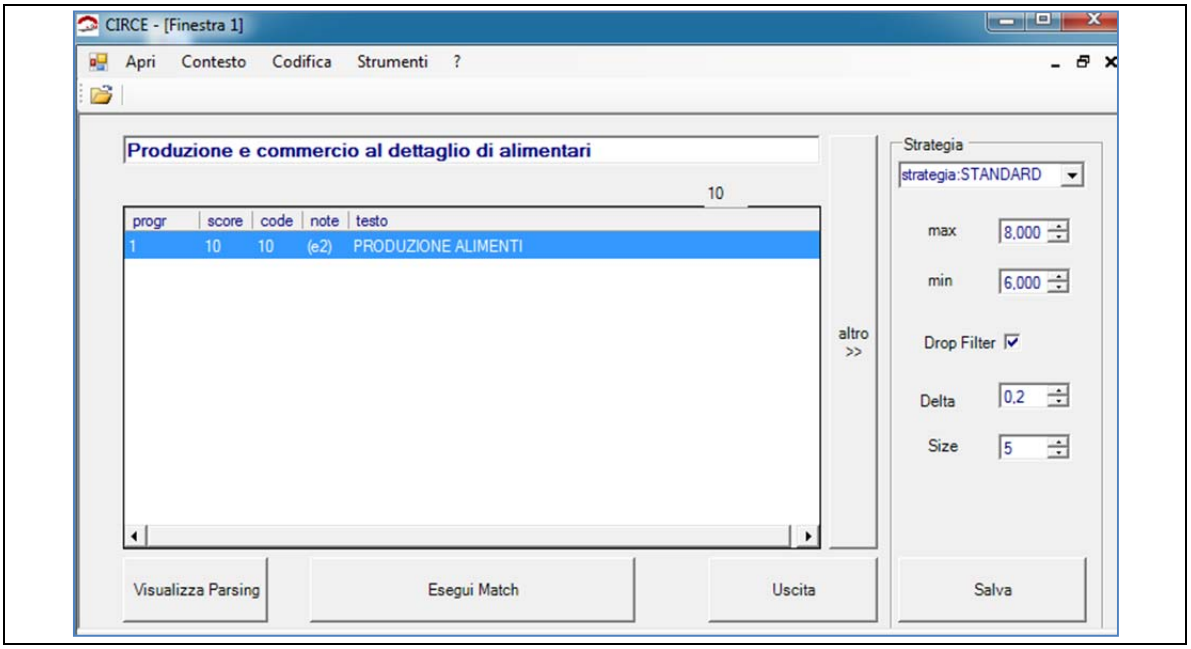
a) concentrare inizialmente l'attenzione sull'Ateco, sfruttandone i "criteri classificatori" attualmente in uso;
b) introdurre le funzioni di somiglianza all'interno di un nuovo passo di standardizzazione delle stringhe di input, preliminare a quelli previsti dalla versione di CIRCE attualmente in esercizio.

La struttura della classificazione Ateco è molto complessa e si basa su una molteplicità di criteri astratti, la cui natura dipende – per altro – dal livello di dettaglio cui ci si riferisce. Per conseguenza, la determinazione automatica del codice Ateco necessita di accorgimenti particolari, quali la definizione e l'esecuzione di regole *ad hoc*. Sono un esempio di regole *ad hoc* i cosiddetti "criteri classificatori" cui si è accennato nel paragrafo precedente. Durante la fase di standardizzazione, CIRCE trasforma ogni stringa di input assoggettandola a trasformazioni successive, effettuate in cascata. Durante tale trasformazione, il software valuta l'applicazione dei "criteri classificatori" disponibili, sulla base della forma corrente della stringa. Operativamente, l'applicazione di uno specifico "criterio classificatorio" scatta qualora la stringa in corso di trasformazione contenga uno specifico insieme di "parole criterio". Insistendo sull'esempio del paragrafo precedente: l'occorrenza (consecutiva, ma in ordine arbitrario) della coppia di parole criterio "produzione" e "vendita" determina l'eliminazione della parola "vendita" in favore della parola prevalente "produzione". Le *parole criterio* si configurano, dunque, come termini "speciali" in grado di guidare il sistema di classificazione automatica verso specifiche divisioni della classificazione di riferimento.

In considerazione del potere discriminatorio peculiare delle parole criterio, si comprende l'effetto potenzialmente catastrofico di eventuali errori ortografici che dovessero impedirne l'identificazione da parte dell'algoritmo di codifica automatica. Le figure 41 e 42 seguenti mostrano i risultati ottenuti da CIRCE nella codifica Ateco di due stringhe che differiscono per un solo

carattere: nella prima compare la parola "commercio", nella seconda l'errore ortografico "commerccio". Il punto chiave dell'esempio è, ovviamente, che il termine "commercio" è per CIRCE una parola criterio.

Figura 41 - Risultato codifica stringa "Produzione e commercio al dettaglio di alimentari"

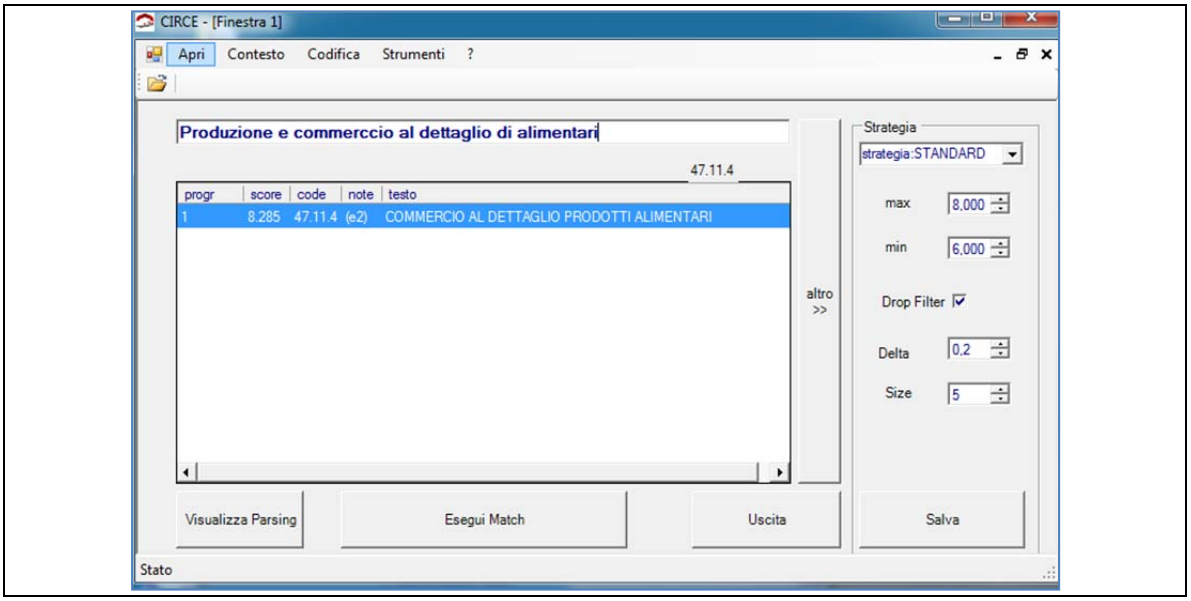


Come si deduce dalla figura 41, il risultato della codifica della prima stringa è un *match* unico, corretto, con punteggio massimo (uguale a 10). La corretta identificazione della parola criterio "commercio" ha, infatti, consentito a CIRCE di applicare tutte le trasformazioni e le prevalenze previste nei file di *parsing* dell'Ateco ed appropriate alla stringa corrente:

1) PRODUZIONE COMMERCIO = PRODUZIONE
2) PRODUZIONE DETTAGLIO = PRODUZIONE

L'esito finale è che l'attività economica è stata codificata correttamente all'interno dell'industria alimentare.

Figura 42 - Risultato codifica stringa "Produzione e commerccio al dettaglio di alimentari"



Nel secondo caso, al contrario, la presenza dell'errore ortografico "commerccio" sulla parola criterio – errore sfortunatamente non previsto nel file dei sinonimi – impedisce a CIRCE di applicare gli adeguati criteri classificatori. La conseguenza è, come mostrato nella figura 42 un falso match. CIRCE associa univocamente, ma erroneamente, la stringa di input al settore del commercio, anziché a quello dell'industria alimentare. Si noti, incidentalmente, come il punteggio attribuito al (falso) *match*, pur inferiore al valore massimo, è piuttosto elevato e dunque fuorviante: utenti non esperti, potrebbero essere indotti a ritenere il risultato della codifica soddisfacente.

La sensibilità di CIRCE agli errori ortografici e/o di digitazione commessi sulle parole criterio configura queste ultime come l'obiettivo ideale su cui incardinare il primo ciclo di sperimentazione delle funzioni di somiglianza. L'idea di base è molto semplice: sfruttare le funzioni di somiglianza fra stringhe per "riconoscere" eventuali parole criterio contenute nella stringa di input, anche qualora esse fossero digitate in modo scorretto. A valle di questo passo preliminare di "riconoscimento automatico" – concentrato, ripetiamo, sulle sole parole criterio – il flusso di lavoro di CIRCE proseguirebbe inalterato, con l'esecuzione degli usuali algoritmi di standardizzazione dei testi e di *matching* fra stringhe.

Tecnicamente, il "modulo di riconoscimento automatico delle parole criterio" confronterebbe le singole parole della stringa di input con le parole criterio, calcolandone le somiglianze, ed, in presenza di valori di somiglianza superiori alla soglia di accettazione, sostituirebbe la parole di input con la corrispondente parola criterio. Nel caso in cui, per una determinata parola di input, dovessero esistere più parole criterio caratterizzate da somiglianze superiori alla soglia, l'algoritmo potrebbe, alternativamente: (i) privilegiare l'efficienza e selezionare la parola criterio incontrata per prima, (ii) privilegiare l'accuratezza e selezionare la parola criterio più somigliante.

Anche tenendo conto dell'intrinseco aumento di complessità computazionale associato alla transizione dal confronto per uguaglianza a quello per somiglianza, la complessità computazionale del modulo di riconoscimento resterebbe certamente gestibile. Infatti, le parole criterio di una classificazione, quand'anche se ne considerino tutti i sinonimi conosciuti, costituiscono un sottoinsieme molto ristretto delle voci di dizionario della classificazione. Per dare un'idea, i criteri classificatori attualmente utilizzati da CIRCE per la codifica dell'Ateco ruotano intorno alle 11 parole criterio primarie riportate nella tavola 23, ed ai relativi sinonimi.

Tavola 23 - Parole criterio del dizionario informatizzato Ateco 2007

Parole criterio Ateco 2007
PRODUZIONE
COMMERCIO
INGROSSO
DETTAGLIO
RIPARAZIONE
MANUTENZIONE
INSTALLAZIONE
ALBERGO
BAR
RISTORANTE
TERME

Come dovrebbe essere ormai evidente, l'obiettivo fondamentale del modulo di riconoscimento proposto sarebbe la "correzione automatica" degli errori ortografici nelle parole criterio eventualmente presenti nelle stringhe di input da codificare. Si tratterebbe, tuttavia, di una soluzione di gran lunga preferibile a quella – cui è stato fatto cenno nel paragrafo precedente – di sottoporre tutti i testi di input ad un programma automatico di controllo ortografico. Infatti:

1) il riconoscimento automatico sarebbe ristretto alle sole parole criterio, dunque l'eventuale correzione si concentrerebbe su pochi errori molto influenti;
2) le scelta della funzione di somiglianza e – soprattutto – della soglia di accettazione potrebbero essere adattata alla classificazione di interesse.

È lecito attendersi che entrambi i fattori (1) e (2) contribuiscano ad attenuare il rischio di "falsi positivi", ovvero *match* erronei derivanti da correzione ortografiche automatiche non adatte al contesto della classificazione.

Vale la pena osservare, inoltre, come gli "inventari" degli errori ortografici più frequenti com-

messi nelle procedure di codifica automatica – inventari che l'Istat manutiene ed aggiorna costantemente per le classificazioni di maggiore interesse – costituiscano una ricchissima base di conoscenza utilizzabile per risolvere il problema (2). Infatti, la funzione di somiglianza s(., .) e la soglia di accettazione s_{th} più adatte alla classificazione di interesse potrebbero essere identificate, in prima approssimazione, massimizzando l'accuratezza della correzioni indotte dalla coppia < s(., .), s_{th} > sugli errori ortografici "presenti in inventario" e relativi alle parole criterio della classificazione.

Concludiamo sottolineando come, dal punto di vista informatico, l'implementazione del modulo di *"riconoscimento automatico delle parole criterio"* qui brevemente illustrato non presenti difficoltà tecniche di particolare rilievo. L'algoritmo descritto è molto semplice ed R, il linguaggio di programmazione adottato per CIRCE, rende disponibile un'ampia scelta di funzioni di somiglianza fra stringhe. È, anzi, possibile affermare che i tre package R 'proxy' (Meyer and Buchta 2015), 'RecordLinkage' (Borg and Sariyar 2015) e 'stringdist' (van der Loo 2014) coprano, complessivamente, l'intero spettro delle funzioni di somiglianza fra stringhe effettivamente rilevanti in ambito applicativo.

7.2 Lo sviluppo dei web service per la codifica on-line e la codifica *batch*

Come già anticipato nel capitolo 4, l'utilizzo della modalità web service, non solo permette di fornire un servizio più funzionale, sicuro ed affidabile rispetto al passato, ma anche e soprattutto un servizio che può essere "riutilizzato" in altri contesti di codifica grazie alle caratteristiche di portabilità e riusabilità della sua architettura.

L'esperienza maturata potrebbe infatti consentire lo sviluppo di servizi analoghi relativi a:

a. altre classificazioni, per la codifica on-line di variabili testuali quali la Professione, il Titolo di studio, il Comune e lo Stato estero;
b. altre modalità di codifica, come quella *batch*, di variabili testuali sopra citate;
c. servizi esterni, che offrano un servizio di codifica on-line e/o *batch* all'esterno dell'Istituto, indipendentemente dalle pagine del sito istituzionale

Trattandosi anche in questo caso di innovazioni, si delineeranno, nei paragrafi seguenti, i vantaggi, gli svantaggi e le precauzioni da adottare nell'adozione dei web service sopra descritti.

7.2.1 Web service interno per la codifica on-line

L'applicazione di codifica web sviluppata per l'Ateco si basa su un web service definito "interno" ossia residente su server accessibile solo attraverso la rete interna dell'Istituto. L'accesso al servizio da parte di utenti esterni alla rete Istat avviene accedendo ad una pagina web dell'Istituto che a sua volta chiama il web service attraverso un componente software client. Sulla pagina web vengono visualizzati i risultati prodotti dal servizio in modalità e struttura accessibile per l'utente finale.

Da un punto di vista tecnico, la disponibilità di un web service facilita notevolmente l'utilizzo del servizio da parte di altre applicazioni informatiche, residenti su server Istat, in quanto è molto più agevole "inglobarne" i risultati offerti. La soluzione web service offre, infatti, numerosi vantaggi rispetto ad altre tipologie di integrazione del software, nello specifico:

1. *Interoperabilità*: il loose coupling è il beneficio più importante dei web services che tipicamente trova la maggiore esplicazione nell'uso di servizi esposti sul web, che offrono ai soggetti interessati allo sviluppo di un'applicazione distribuita, sia essa web based o meno, la possibilità di integrare in modo agevole componenti di terze parti. I web services, a fronte dell'incremento di interoperabilità software,possiedono normalmente un ciclo di vita più lungo rispetto a componenti software tradizionali (esempio librerie software) offrendo una maggiore e migliore diffusione. Dato che i web services utilizzano metodi standard di comunicazione in rete (http su tcp/ip) offrono una completa indipendenza dalla piattaforma di destinazione: nessuna dipendenza dal linguaggio di programmazione utilizzato per creare l'applicazione distribuita e dal sistema operativo dell'applicazione ospite. I web services sono dunque indipendenti dalla piattaforma ed orientati ad una filosofia di massima diffusione.

2. *Usabilità*: I web services consentono il raggiungimento di un maggiore livello di usabilità del software, perché chi progetta e realizza un'applicazione può farlo integrando in modo trasparente il servizio e strutturando l'applicazione finale in funzione delle proprie esigenze, piuttosto che riscrivere la logica del servizio o essere vincolato ad una particolare forma di comunicazione o memorizzazione dei dati. Questo fattore è di primaria importanza nella diffusione del servizio, in quanto svincola il potenziale utente dal dover installare nuovo software.
3. *Riusabilità*: l'approccio web service semplifica il riutilizzo del software; non è necessario implementare di nuovo le funzionalità offerte in altre applicazioni, ma basta semplicemente integrare il servizio, superando così le problematiche di manutenzione del software e delle sue evoluzioni, nonchè gli aspetti di legacy o di software proprietario che chi sviluppa il componente deve gestire (licenze d'uso ad esempio).
4. *Rilascio* (Deployability): I web services sono soggetti ad un rilascio tramite tecnologia standard web, questo fatto rende possibile aggirare le usuali limitazioni dall'esterno verso la rete corporate, quindi il servizio può essere esposto senza alterare le politiche di sicurezza e di gestione dei firewall.
5. *Flessibilità* operativa: importante è anche la possibilità di poter predisporre una strategia di sicurezza (trust policy) attraverso l'utilizzo del protocollo HTTPS (canale sicuro basato su meccanismi di crittografia a chiave pubblica e certificati di sicurezza). L'eventuale possibilità di effettuare un filtro sugli indirizzi IP (IP filtering) delle richieste in arrivo e/o l'utilizzo di meccanismi di Client-Authentication, permette inoltre di gestire in maniera puntuale l'accesso ai sistemi da parte degli utilizzatori esterni del servizio.

Tuttavia, a fronte dei vantaggi, vanno anche discusse le potenziali problematiche relative all'uso di web services:

1. *Inefficienza*: dato che i web services si basano sullo scambio di contenuti sostanzialmente testuali tramite HTTP/HTTPS, si ha un'efficienza ridotta rispetto a soluzioni che implementano un interscambio binario dei dati, specie se locale (localhost) o relativo ad una rete corporate (rete locale). Tuttavia, in situazioni analoghe a quelle della codifica dell'Ateco, questo problema è sostanzialmente irrilevante in quanto i messaggi SOAP scambiati hanno dimensione molto contenuta e le richieste al servizio di codifica sono di tipo stateless, ossia non devono affrontare il problema della gestione di una ipotetica sessione. Si tratta, inoltre, di servizi 'sincroni' e 'one-shot' , cioè ad una richiesta segue una risposta, senza scambio di una molteplicità di messaggi bidirezionali che potrebbero avere problemi di latenza ed inefficienza in caso di connessioni lente o congestione della rete su Internet.
2. *Gestione di sessioni*: i web services sono intrinsecamente privi di un supporto nativo di gestione delle sessioni e connessioni stateful di lungo periodo, differentemente da altre tecnologie di integrazione quali ad es. CORBA o RMI.
3. *Gestione dei token* di sessione e degli errori lato client: nella gestione di sessioni i web servers si basano sul rilascio di token di sessione ai client (tipicamente nella forma di cookies di sessione) per mantenere un ambito stateful e tracciare le sessioni utente nel tempo. Se il web service è interessato a mantenere informazioni sullo stato dei client, deve comunicare periodicamente con essi, gestire eventuali interruzioni (timeouts) e situazioni anomale (crash dei client e conseguente mancanza di comunicazione con essi), effettuare rollbacks a cascata. Comunque, nel caso di web services analoghi a quello dell'Ateco, che sono stateless e non utilizzano transazioni, non si tiene traccia dello stato dei client e dunque questa problematica è irrilevante.

Da quanto esposto si evince che, nel caso in esame, il sistema con web service presenta solo vantaggi, mentre le situazioni svantaggiose non possono presentarsi per definizione oppure sono limitate a situazioni transitorie e contingenti (es. congestione o connessioni lente).

È opportuno precisare che in generale, un web service, per garantire la sicurezza delle comunicazioni, si serve di tecniche basate sul messaggio piuttosto che sul trasporto.

Fra le minacce che affliggono la sicurezza dei flussi comunicativi ricordiamo:

- alterazione del messaggio: compromissione dell'integrità del messaggio per modifica/ cancellazione/aggiunta di parti;
- confidenzialità: accesso non autorizzato all'informazione;
- Man-in-the-Middle: l'attaccante intercetta il messaggio fra il richiedente e il ricevente;
- spoofing: l'attaccante assume un'identità fidata compromettendo la trust relationship fra gli interlocutori;
- replay attacks: si intercetta il messaggio e si rinvia per farlo processare nuovamente.

L'utilizzo di web service prevede inoltre la possibilità di usufruire di WS-Security ossia una specifica in aggiunta al messaging SOAP che permette di garantire l'integrità e la confidenzialità dei messaggi scambiati.

Sotto il profilo tecnico, per l'utilizzo del web server in varie applicazioni l'unica operazione rilevante è rappresentata dalla programmazione di un client per il web service. Dato che le tecnologie web di punta (php, MS .Net, Java,Python, Ruby on Rails) e tutti i maggiori framework di sviluppo di applicazioni possiedono librerie, comandi e strutture dati standard per l'utilizzo di web services, nello specifico basati sul protocollo SOAP, lo sviluppo delle applicazioni risulterebbe semplificato ed il carico del lavoro di integrazione per gli sviluppatori sarebbe sostanzialmente ridotto all'essenziale.

Un valido esempio di riutilizzo di quanto realizzato per l'Ateco, consisterebbe nel far si che i questionari per le indagini web possano richiamare un servizio per la codifica di altre variabili. Il web service sviluppato per l'attività economica potrebbe quindi essere duplicato modificando soltanto il sistema di codifica sottostante, in questo caso CIRCE, nel quale verrebbero inseriti i dizionari della variabile d'interesse, insieme ai file contenenti le regole di standardizzazione e codifica dei testi propri di ciascuna classificazione.

Quanto all'utilizzo del web service, le applicazioni relative alle raccolta dati via web potrebbero, ad esempio, richiamare il servizio qualora prevedessero la rilevazione e la codifica on-line di variabili testuali che afferiscono ad una classificazione ufficiale. Un caso concreto è rappresentato dalle indagini sulle imprese alcune delle quali richiamavano la vecchia applicazione web di codifica dell'Ateco per agevolare il rispondente nell'individuazione del codice dell'attività economica dichiarata. L'uso della modalità web service non solo permetterebbe un accesso più semplice all'applicazione di codifica, ma offrirebbe anche una modalità standard di accesso alle funzionalità che eviterebbe, a parità di piattaforma e linguaggio di programmazione, la riscrittura di codice ad hoc per ogni questionario d'indagine in funzione del prodotto/linguaggio con cui è stato sviluppato.

L'utilizzo di web service di codifica, in fase di raccolta dati, inoltre aiuterebbe i rispondenti a non fornire risposte generiche a domande di difficile rilevazione, come per esempio la Professione o il codice Ateco. Il lavoro a posteriori sarebbe quindi ridotto e finalizzato a codificare esclusivamente le stringhe alle quali non è stato possibile attribuire un codice univoco con la procedura on-line. Tutto ciò con un grosso risparmio sia in termini di tempo che di risorse.

Accanto a questi vantaggi, occorre aggiungere che la semplificazione tecnica di utilizzo del servizio di codifica potrebbe essere uno sprone alla progettazione di test finalizzati ad individuare il sistema di codifica on-line più idoneo da usare nelle indagini web, differenziandolo per target di indagine (popolazione e aziende) e per variabile di interesse. Esistono infatti diverse tipologie di sistemi di codifica, che ricercano *match* esatti o parziali, basati a loro volta su diversi algoritmi di *match* (*dictionary algoritms, weigthing algortims,* ecc.) sulla cui adozione influisce soprattutto la struttura e complessità della classificazione. A questi si possono associare diversi sistemi di "navigazione" della base informativa (per ramo -possibile solo per classificazioni gerarchiche-, per dizione o mista) da fornire all'utente del sistema di codifica.

CIRCE appartiene alla categoria dei *weighting algorithms* e ricerca *match* esatti o parziali sulla base del peso delle parole ossia del loro grado di informatività. Questo fa si che per ottimizzare la ricerca delle stringhe e quindi la loro codifica sia necessario che i testi inseriti siano scritti secondo alcune regole (stringhe non generiche, non eccessivamente lunghe, uso di parole chiave). Nel contesto delle indagini assistite da computer, queste regole possono essere inserite fra le istruzioni alla compilazione del questionario elettronico, facendo però attenzione ad non incrementare il fastidio statistico per il rispondente vanificando il vantaggio intrinseco dell'applicazione on-line.

In Istat esistono altri strumenti di supporto alla codifica on-line, come il Navigatore delle Pro-

fessioni (http://cp2011.istat.it/), basato sui *dictionary algorithms*, che ricerca le stringhe effettuando accoppiamenti esatti sulla base di una o più parole, permettendo di navigare dentro la struttura della classificazione con una ricerca per ramo oppure per dizione. Anche questo sistema, alla stessa stregua di CIRCE, presenta alcuni aspetti che potrebbero creare del fastidio statistico nel rispondente all'indagine web. Il suo utilizzo infatti richiede un certo livello di conoscenza della classificazione, che in generale non si riscontra fra i rispondenti, rendendolo più adatto per indagini con rilevatore, appositamente istruito, piuttosto che per quelle auto-somministrate.

Sarebbe interessante mettere a confronto i due sistemi per capire quale dei due garantisca risultati più incoraggianti in termini sia di qualità che di tassi della codifica, arrecando al contempo il minor fastidio statistico al rispondente. Ciò sarebbe fattibile, come spiegato in precedenza, prevedendo una molteplicità di funzioni di codifica differenti nel web service e mantenendo una sostanziale similitudine ed uniformità nell'interfaccia di accesso a motori di codifica potenzialmente (molto) differenti tra loro che consentirebbe un più confronto più agevole permettendo inoltre l'ottimizzazione del patrimonio software in tema di codifica automatica e/o assistita.

In questo contesto è evidente che l'utilizzo di un web service rappresenta la strategia ideale per sopperire alle differenti tecnologie, piattaforme, strategie implementative potenzialmente coinvolte nel medio-lungo periodo anche a seguito della rapida evoluzione dell'informatica a compendio della evoluzione, pur presente, dei metodi statistici ed ingegneristici in senso lato.

7.2.2 Web service interno per la codifica batch

ACTR v3, prima, e CIRCE, oggi, sono sistemi ufficiali utilizzati in Istat per la codifica *batch*, ossia a posteriori, delle variabili a testo libero afferenti a classificazioni ufficiali e rilevate nelle indagini. Con l'adozione di ACTR v3 nel 1998 è stato possibile ridurre drasticamente l'attività manuale di codifica, con notevoli vantaggi in termini di tempestività del rilascio dei dati nonché di incremento della qualità della codifica a seguito della standardizzazione del processo stesso; con la codifica automatica è stata infatti eleminata l'influenza sui risultati della preparazione e soggettività interpretativa del codificatore manuale.

Ovviamente la "componente" umana gioca un ruolo importante anche nel processo di codifica automatica in quanto è vitale per controllare la qualità del sistema e per addestrare lo stesso attraverso l'analisi dei risultati.

Per valorizzare i vantaggi offerti da un potenziale web service interno è necessario tener conto, come dettagliatamente descritto nel capitolo 2, delle varie fasi di gestione e di realizzazione della codifica *batch* che avviene in stretta collaborazione l'unità referente del software di codifica (attività centralizzata) e le unità responsabili delle varie classificazioni (settori di produzione).

I settori di produzione, che necessitano della codifica dei testi rilevati durante le indagini, inviano infatti un file contenente le stringhe da codificare ed ottengono come output i file associati ai possibili esiti della codifica (un file dei testi codificati in modo unico, un file dei testi associabili a diversi codici, un file dei testi non codificati).

L'unità referente del software di codifica, per migliorare i tassi di codifica e la qualità dei codici assegnati, dopo il primo passaggio di codifica realizza diverse attività, che consistono nell'analisi dei casi non codificati per capire ed eventualmente eliminare le cause del mancato abbinamento. L'analisi dei testi non codificati permette quindi di "addestrare" il sistema, aggiungendo sia dizioni mancanti sia nuovi sinonimi; a questa fase segue un nuovo passaggio di codifica, che mostra, generalmente, una riduzione dei "falliti" e quindi un contemporaneo aumento della percentuale di testi codificati.

Per ogni "passaggio" di codifica viene misurata la qualità dei risultati per individuare gli eventuali falsi *match*.

Le attività sopra descritte prevedono l'intervento sia dell'esperto del software che dell'esperto della codifica che deve verificare che ogni proposta di integrazione del dizionario o dei file di *parsing* non alteri la struttura della classificazione apportando distorsioni all'intero processo di codifica.

Dopo questi passaggi, il file viene riconsegnato all'unità di produzione accompagnato anche da una relazione che descrive la fase di codifica, le modifiche introdotte nel sistema a seguito dell'analisi dei non *match* e dei *match* multipli, il tasso di codifica e la qualità raggiunta.

Da quanto descritto, emerge chiaramente il vantaggio della "gestione centralizzata" del sistema

di codifica: la fase di addestramento conseguente ogni richiesta di codifica *batch* è in realtà un modo per aggiornare continuamente il dizionario ed il contesto di codifica, affinando le regole di attribuzione dei codici e arricchendo i testi inseriti con la variabilità del linguaggio umano. In questo modo si migliorano i risultati della codifica non solo per il singolo file di testi, ma per tutti coloro che necessitano di codificare la stessa variabile.

I vantaggi, in termini di miglioramento dei tassi e della qualità della codifica che si ottengono da una gestione centralizzata della codifica *batch*, potrebbero essere ampliati se si allargasse "la rosa" degli utenti della codifica *batch*, trasformando l'attività in un web service. In modo analogo a quanto realizzato per la codifica on-line, potrebbe essere implementato un web service per la codifica *batch*, il quale, invece di codificare singole stringhe di testo e mostrare i risultati a video, come avviene per la codifica on-line, permetterebbe di codificare interi file di testi e di memorizzare i risultati in appositi file disponibili all'utente.

In questo modo si darebbe a molti più utenti la possibilità di codificare i testi derivanti da indagini o da altre fonti, senza la necessità di attendere che la propria richiesta sia evasa dal servizio responsabile della codifica. A questo si aggiungerebbe anche il fatto che una platea più vasta di utenti permetterebbe di arricchire il dizionario con termini e modi di dire usati da target diversi come famiglie, aziende, amministrazioni pubbliche ecc. Come conseguenza si offrirebbe uno standard procedurale per la codifica *batch* che, grazie all'aggiornamento del dizionario tramite diverse fonti/utenti, garantirebbe livelli di qualità sempre crescenti (come già avviene del resto, ma con un'accelerazione maggiore).

L'aggiornamento del dizionario e degli ambienti di codifica continuerebbe ad essere gestito in modo centralizzato dagli esperti del sistema di codifica e delle classificazioni ai quali i singoli utenti potrebbero rivolgersi qualora avessero bisogno di ulteriori passaggi di codifica e quindi dell'analisi dei risultati ottenuti dal servizio usato in maniera autonoma.

Per garantire la centralità dell'addestramento, l'architettura del web service dovrebbe essere realizzata in modo tale che si tenga memoria dei risultati della codifica. I risultati (file degli unici, multipli, possibili e falliti) dovrebbero quindi essere accessibili sia all'unità che usa il servizio sia a gestori del servizio stesso che potrebbero in questo modo analizzarli ed introdurre i miglioramenti da questi richiesti al sistema di codifica.

Eseguita la fase di addestramento, potrebbe essere inviata una notifica agli utenti che indica l'avvenuto aggiornamento del sistema, disponibile ad essere "chiamato" di nuovo per ottenere una codifica dai risultati potenzialmente migliori.

7.2.3 Un'ipotesi architetturale di web service interno per la codifica batch

Il sistema attuale non è progettato per svolgere funzioni di codifica *batch*, tuttavia per un possibile sviluppo futuro l'architettura del sistema potrebbe essere rivista per consentire la gestione di elaborazioni *batch*, anche di lungo periodo, e rendere disponibile agli utenti la sequenza dettagliata delle stringhe immesse e dei risultati prodotti. Il sistema di codifica *batch* necessiterebbe, inoltre, di estensioni software e di ulteriore hardware dedicato nonché di risorse, in termini di spazio disco, per tenere traccia dell'esito delle richieste *batch*.

Si presentano qui a riguardo alcune tematiche da approfondire nell'eventualità di una progettazione di dettaglio architetturale ed implementativa correlata alle nuove funzionalità *batch*.

Considerate le limitazioni dell'interfaccia web service rispetto a sessioni costituite da una numerosa molteplicità di richieste e considerata la natura dei processi in questione (*batch*) che non prevedono una interazione immediata con l'utente, una volta lanciato il processo di elaborazione si opterebbe per una possibile soluzione come di seguito descritta, a meno di varianti derivanti da considerazioni successive:

1. La richiesta *batch* avverrebbe tramite la usuale interfaccia web service, opportunamente integrata di una funzione di codifica addizionale per i processi *batch* in cui l'utente specificherebbe il tipo di codifica da utilizzare, uno o più indirizzi email, ed un file, il cui formato sarebbe da definire (es. csv, xml, plain-text o altro), che verrebbe caricato sul server possibilmente in memoria, evitando di scrivere sul filesystem per non violare le policy di sicurezza già esposte nel capitolo 4.

2. Il sistema, nello specifico il web service, provvederebbe ad assegnare alla richiesta un identificativo ed altri dati di gestione ed a smistare i dati contenenti le stringhe da codificare verso l'application server ed il motore di codifica, che dovrebbe essere preferibilmente mutithreading e dotato delle risorse hardware necessarie allo smaltimento delle richieste *batch* in tempi ragionevoli a fronte del carico previsto (da definire in sede di studio di fattibilità ed analisi).
3. Le richieste concorrenti degli utenti, sia online che *batch*, verrebbero gestite nella stessa coda attraverso i meccanismi di parallelismo garantiti dal web server Apache e del suo pool di esecuzione, idem per il logging lato application server.
4. Il motore di codifica scelto si occuperebbe di gestire e ritornare le singole codifiche in un formato convenzionale come JSON. Il JSON risultante in ASCII potrebbe essere compresso nel formato gzip per non appesantire eccessivamente le comunicazioni tra l'application server ed il backend server[22].
5. Il web service nell'applicazione di backend (WITCH), dopo aver decompresso il file ricevuto (sempre in memoria onde non coinvolgere il filesystem sul backend server) fornirebbe come risposta il *parsing* del file ASCII risultante e del JSON contenente le *query* e le codifiche. Infine, per storicizzare le richieste e tenerne traccia, memorizzerebbe alcuni dati sul DB: risultato, timestamp, dettaglio dell'engine di codifica utilizzato, codice univoco.
6. L'ulteriore step consisterebbe nella costruzione di un file intelligibile per l'utente finale contenente le stringhe ricercate ed i relativi risultati prodotti dalla procedura di codifica, ossia un report da restituire all'utente tramite email in formato testo compresso (zip). Come ulteriore funzionalità avanzata, per utenti e/o servizi dotati delle opportune risorse tecniche, si potrebbe fornire la possibilità di accedere in lettura ad una porzione del database facendo utilizzare il codice univoco (inviato dal sistema per email all'atto di presa in carico della richiesta *batch*) per individuare il dataset di risposta; questa funzionalità potrebbe essere utile per consentire all'utente di incrociare i dati di richieste differenti estraendo le informazioni direttamente dal database.

Ovviamente il sistema di codifica, l'applicazione di backend e i motori di codifica andrebbero opportunamente modificati per consentire il carico computazionale ed i tempi di attesa relativi all'elaborazione di richieste di codifica *batch* impegnative e concorrenti.

La gestione intrinseca delle code produrrebbe un accodamento automatico delle richieste concorrenti; per le considerazioni svolte non è indifferente considerare l'utilizzo di un application server (o di un cluster di application servers) dedicati alle sole codifiche *batch*, per non interferire pesantemente con le operazioni di codifica online rischiando di peggiorare sensibilmente i tempi di risposta.

7.2.4 Web service esterni per la codifica on-line o batch

I due servizi interni di codifica on-line e *batch* descritti nei due paragrafi precedenti potrebbero essere rivolti all'esterno dell'Istituto, raggiugendo un numero elevato di tipologie di utenti nell'ambito del territorio nazionale (il Sistan, il mondo accademico, gli Enti pubblici, i singoli cittadini). In questo paragrafo saranno descritti i vantaggi, gli svantaggi e le precauzioni tecniche necessarie ad offrire un servizio "sicuro" ed "efficiente" in termini di prestazioni e di risultati di codifica.

Dal punto di vista della diffusione del dato statistico i due servizi potrebbero essere di supporto alla divulgazione della conoscenza e dell'uso delle classificazioni ufficiali che viene già realizzata dall'Istat attraverso la pubblicazione sul sito istituzionale del "Sistema delle Classificazioni" (http://www.istat.it/it/strumenti/definizioni-e-classificazioni) dove è possibile trovare tutte le classificazioni ufficiali adottate dall'Istituto, navigare all'interno della loro struttura, conoscere le defini-

[22] Attualmente tale comunicazione avviene in loopback, ma non è detto che, a fronte di considerazioni prestazionali, nuove implementazioni di strumenti di engine di codifica risiederanno sullo stesso application server utilizzato per CIRCE .

zioni adottate ed i codici attributi alle singoli "voci" elementari che le compongono.

Dal punto di vista tecnico, nell'eventualità di rilasciare il servizio di codifica *batch* all'esterno, andrebbero tenute in particolare considerazione le problematiche addizionali di performance e di ottimizzazione. Infatti, l'appesantimento dovuto al carico computazionale extra, potrebbe prestarsi, in assenza di policy di sicurezza addizionali e restrittive, a tipologie di attacco informatico (DoS). Per queste considerazioni si rimanda a studi più dettagliati da effettuare nel caso in cui la necessità di implementare un servizio *batch* fruibile da esterno divenisse necessaria. Un'alternativa potrebbe consistere nella serializzazione delle sole richieste *batch*, attuando una logica operativa diversa (seriale invece che concorrente) nella funzione di codifica *batch*, con accodamento delle richieste utente, visto che i processi *batch* non hanno le stesse esigenze temporali delle codifiche online.

La disponibilità di servizi esterni di codifica permetterebbe agli utenti di usarli nelle loro applicazioni ed all'Istat di avere a disposizione dati utili a portare avanti un altro obiettivo che si propone il "Sistema delle Classificazioni" che è quello di armonizzare le definizioni relative a stesse variabili, ma usate o rilevate da soggetti diversi e con finalità diverse da quelle statistiche (fini fiscali, amministrativi, ecc.). Questo, tra l'altro, potrebbe essere di ausilio alla realizzazione del nuovo scenario della statistica ufficiale sempre più impegnata ad integrare dati provenienti da fonti amministrative al fine di ottimizzare i tempi di rilascio delle informazioni, aumentando la qualità e riducendo le ridondanze informative e quindi riducendo l'impegno di risorse umane ed economiche.

Per ottenere questi vantaggi sarebbe però necessario che i dati codificati dai diversi utenti fossero resi disponibili all'Istat al fine di analizzarli e procedere all'arricchimento ed addestramento delle basi informative che, come già detto, è condizione necessaria per il miglioramento sia del recall che del precison rate. Questa attività avrebbe un impatto di un certo rilievo sul lavoro del personale responsabile del software e delle classificazioni, e richiederebbe quindi un potenziamento ed una riprogettazione di come queste attività sono attualmente organizzate in Istituto.

Riferimenti bibliografici

Amdahl, G. M. 1967. *Validity of the single processor approach to achieving large scale computing capabilities.* In AFIPS Spring Joint Computing Conference, AFIPS/ACM/Thomson Book Company, Washington D.C., pp. 483-485.

Arpi A. 2015. *Usabilità: 6 tips per migliorarsi*. http://www.webhouseit.com/usabilita-6-tips-migliorarsi/.

Borg, A e M., Sariyar. 2015. *RecordLinkage: Record Linkage in R. R package version 0.4-8*. URL https://CRAN.R-project.org/package=RecordLinkage

Carasso, D. 2012. Exploring Splunk. https://www.splunk.com/web_assets/v5/book/Exploring_Splunk.pdf

D'Orazio M., S., Macchia, S. 2002. *A system to monitor the quality of automated coding of textual answers to open questions*. Research In Official Statistics ROS, n.2

Decreto del Presidente della Repubblica, 1 marzo 2005, n. 75. Regolamento di attuazione della Legge 9 gennaio 2004, n. 4 per favorire l'accesso dei soggetti disabili agli strumenti informatici.

Decreto legislativo 7 marzo 2005, n. 82. Codice dell'Amministrazione Digitale.

Decreto Ministeriale 8 luglio 2005. Requisiti tecnici e i diversi livelli per l'accessibilità agli strumenti informatici.

Dipartimento della Funzione pubblica, 2011. *Linee guida per i siti web delle PA*.

Dipartimento della Funzione pubblica, 2014. *Il Protocollo eGLU 2.0. Come realizzare test di usabilità semplificati per i siti web delle PA.*

Direttiva n. 8 del 2009 del Ministro per la pubblica amministrazione e l'innovazione.

Eldad, E. 2005. *Reversing: Secrets of Reverse Engineering*. Wiley Publishing.

Ferrillo, A. et al. 2012. *La funzione su web per l'individuazione del codice ATECO sulla base di una descrizione sintetica e monitoraggio delle performance.* Collana Working Paper Istat (n.4/2012).

Ferrillo, A. 2004. *Codifica automatica della variabile Ateco – Metodologia per l'aggiornamento dell'ambiente di codifica ACTR rispetto alla classificazione delle attività economiche Ateco 2002.* Istat documento interno.

Halili, E.ly H. – Apache JMeter - http://dl.ftl.vn/share/tool/Jmetter/apache-jmeter.pdf - PACKT publishing

http://clusit.it/rapportoclusit/

http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigation-report-2015_en_xg.pdf

https://nvd.nist.gov/

http://nist.gov/

http://www.owasp.org/index.php/Category:OWASP_Tools_Project

https://portswigger.net/burp/

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

Istat, 2014. *Piano di accessibilità dell'Istat per il 2014*. Roma.

Kennedy, D., J. O'Gorman, D. Kearns e M. Ahoroni. 2011. *Metasploit: The Penetration Tester's Guide, No Starch Press.*

Legge 9 gennaio 2004, n. 4. *Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici.*

Macchia, S et al. 2007. *Metodi e software per la codifica automatica e assistita dei dati.* Collana Tecniche e strumenti Istat (n.4/2007).

McCLure, S. e G. Kurtz e J. Scambray. 2013. *Hacker 7.0,* Apogeo.

Meyer, D. and C., Buchta. 2015. *Proxy: Distance and Similarity Measures. R package version 0.4-15.* URL https://CRAN.R-project.org/package=proxy

Mitchell, L.J.2013. *php Web Services, APIs for the Modern Web*. O'Reilly Media.

Polillo, R. 2010. Facile da usare – Una moderna introduzione alla ingegneria dell'usabilità. Apogo.

R Core Team. 2015. R*: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Signore O. 2004. *Le linee guida del W3C per l'accessibilità: evoluzione nella continuità*. Ufficio Italiano W3C presso il CNR, Pisa.

Splunk. *QuickReferenceGuide*. https://www.splunk.com/web_assets/pdfs/secure/Splunk_Quick_Reference_Guide.pdf

Stuttard, D. e M. Pinto. 2011. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws 2ed.* Wiley Publishing.

Thomas, E. 2007. SOA: Principles of Service Design, 1st edition, Prentice Hall Ethan Cerami (2002), Web Services Essentials, 1st edition , O'Reilly.

User's Manual - http://jmeter.apache.org/usermanual/index.html

van der Loo, M. 2014. *The stringdist package for approximate string matching.* The R Journal, 6, pp. 111-122. URL http://CRAN.R-project.org/package=stringdist

Vicari, P. et al. 2009. *L'ambiente di codifica automatica dell'ATECO 2007*. Collana Metodi e Norme Istat (n.41/2009).

Wenzowski M. J. 1988. *ACTR – A Generalized Automated Coding System.* Survey Methodology, vol. 14.

Wickham, H. 2014. *Advanced R*. New York: Chapman & Hall/CRC The R Series.

W3C, 2008. *Web Content Accessibility Guidelines (WCAG) 2.0.*

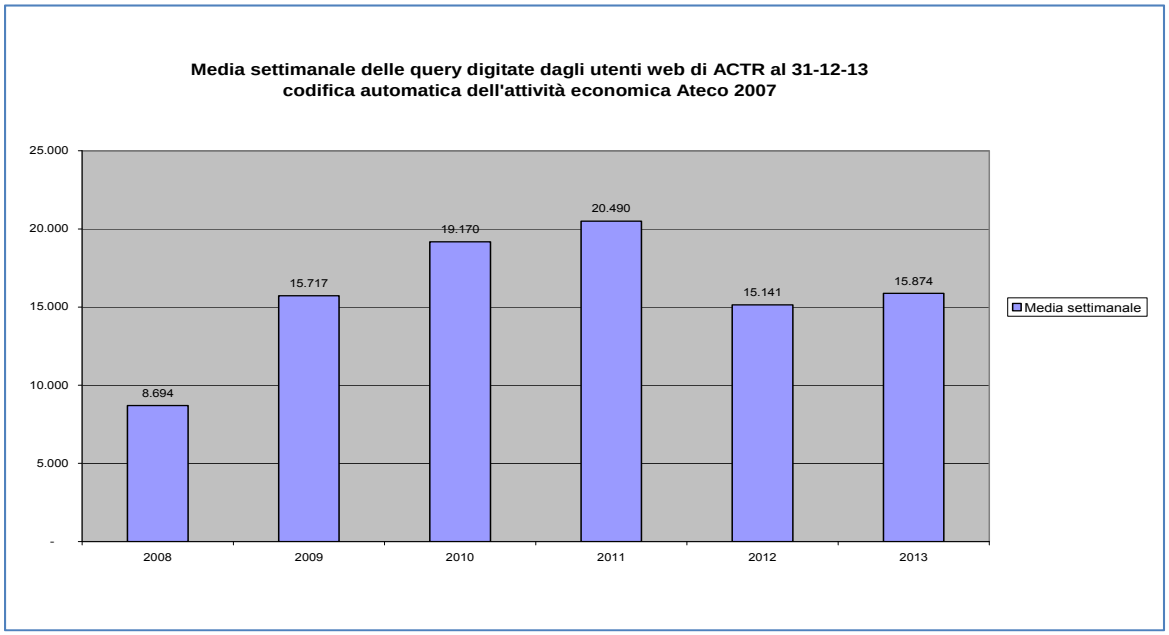
Allegato 1

Angelina Ferrillo
DICA/DCCR/REG/A
30/5/2014

1. ACTR – Codifica automatica attività economica Ateco 2007- Resoconto aggiornamento Actr al 31/12/2013

1. Grafici sulla situazione delle query digitate dagli utenti web.
2. Tabella riepilogativa della codifica automatica delle query prima e dopo l'aggiornamento dell'ambiente di codifica.
3. Tabella degli unici per punteggio (prima e dopo aggiornamento applicazione)

Figura 1 – Media settimanale delle query digitate dagli utenti web di ACTR al 31-12-2013



Dopo il calo relativo all'anno 2012, nel 2013 si è registrato un lieve aumento degli utenti che hanno utilizzato ACTR web per la codifica automatica dell'attività economica.

Figura 2 - Query digitate dagli utenti web per anni e mesi

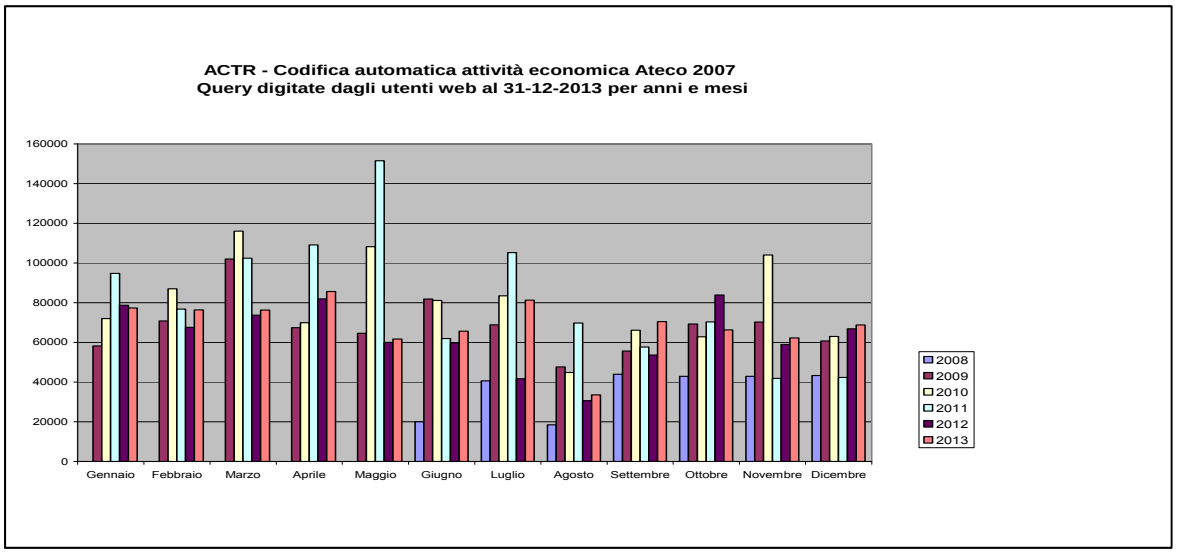


Tavola 1 – Codifica automatica attività economica Ateco 2007 - Situazione aggiornamento ACTR 2012-2013 Confronto risultati codifica con applicazione prima e dopo aggiornamento ambiente di codifica

Tipo di codifica	ACTR Web				ACTR indagini			
	Applicazione dicembre 2012		Applicazione dicembre 2013		Applicazione dicembre 2012		Applicazione dicembre 2013	
	Numero di descrizioni originali	%	Numero di descrizioni originali	%	Numero di descrizioni originali	%	Numero di descrizioni originali	%
Unici	1.764.890	47,10%	2.149.105	47,10%	2.080.810	55,60%	2.517.600	55,10%
di cui Unici con codice ateco n.c. e/o <5 digit	*213.317*	*5,70%*	*262.918*	5,80%	*465.975*	*12,40%*	*561.483*	12,30%
Multipli	232.211	6,20%	285.417	6,20%	90.933	2,40%	117.842	2,60%
Multipli con 5 codici ateco uguali	4.675	0,10%	6.158	0,10%	3.638	0,10%	4.765	0,10%
Possibili	1.299.344	34,70%	1.602.994	35,10%	1.149.734	30,70%	1.433.236	31,40%
Possibili con 5 codici ateco uguali	24.430	0,70%	27.211	0,60%	23.903	0,60%	26.408	0,60%
Falliti	417.916	11,20%	496.118	10,90%	394.448	10,50%	467.152	10,20%
Totale	3.743.466	100,00%	4.567.003	100,00%	3.743.466	100,00%	4.567.003	100,00 %

La tabella permette di confrontare i risultati dell'applicazione di codifica web nel 2012 e 2013 e di avere, per lo stesso periodo, anche un riscontro di qualità ed efficacia del sistema di codifica delle indagini. Quest'ultima è finalizzata a massimizzare la percentuale di codici unici (da cui le percentuali più alte mostrate in tabella), mentre l'applicazione web è finalizzata ad ottenere una quota più cospicua di codici multipli e possibili che rappresentano un ausilio per l'utente al quale viene offerto un ventaglio di codici Ateco tra cui selezionare quello pertinente con la propria attività economica (da questo derivano le percentuali più alte di multipli e possibili per l'applicazione web).

Relativamente all'applicazione ACTR Web emerge che:

– nel 2013 i casi in cui l'applicazione riesce ad individuare un codice corrispondente alla descrizione digitata oppure produce un ventaglio di possibilità (unici + multipli + possibili) sono pari all'89,1% delle descrizioni da codificare, a fronte dell'88,8% registrato nel 2012; di questi la percentuale di unici al 2013 è del 47,1%, invariata rispetto al 2012.
  Occorre sottolineare però che a fronte dell'invarianza della percentuale di unici si ha un

aumento di qualità degli stessi nel 2013. Questo perché sono state sanate delle situazioni in cui il sistema attribuiva un codice unico a descrizioni non univoche. Esempio "impianti elettrici" (frase con una frequenza di 6090) con l'applicazione del 2012 veniva attribuito un codice unico, mentre con questa ultima applicazione viene correttamente codificata come un multiplo.

Inoltre, è stata sanata tutta la parte delle descrizioni riguardanti la nautica (barche, imbarcazioni, navi ecc.), cui spesso veniva attribuito un codice errato.

– la percentuale di falliti nel 2013 è pari al 10,9% a fronte del 11,2% registrato nel 2012, corrispondente ad un decremento dello 0,3%.

Analoghe considerazioni possono farsi anche relativamente all'applicazione ACTR indagini, per la quale il tasso di codifica degli unici risulta in decremento dal 2012 al 2013 dello 0,5%, a fronte però di una qualità migliore.

Esempio "commercio di imbarcazioni" (senza la specifica di ingrosso o dettaglio), prima dell'aggiornamento veniva classificato erroneamente negli unici con il codice della "fabbricazione - 30.12.0". Dopo gli aggiornamenti viene codificato correttamente come un multiplo proponendo il codice di ingrosso e dettaglio.

La migliore qualità dell'applicazione 2013 è confermata anche dall'analisi degli unici per punteggio come risulta dalle tabelle sottostanti.

Le elaborazioni sono state effettuate con le query al 31-12-13, sottoponendole prima ad ACTR aggiornato al 31-12-2012, poi ad ACTR di maggio 2014. La procedura di codifica è quella di ACTR indagini.

Tavola 2 - Unici per punteggio (peso) con applicazione del 31-12-12 (ACTR indagini)

Tipo di codifica: Unici	Peso	Numero di descrizioni originali	%
	8	580.432	23,20%
	9	57.372	2,30%
	10	1.864.123	74,50%
Totale		2.501.927	100,00%

Tavola 3 - Unici per punteggio (peso) con ultima applicazione di maggio 2014 (ACTR indagini)

Tipo di codifica:. Unici	Peso	Numero di descrizioni originali	%
	8	571.756	22,70%
	9	57.095	2,30%
	10	1.888.749	75,00%
Totale		2.517.600	100,00%

Come si può osservare dalle tabelle la percentuale di descrizioni con punteggio 10 registra un incremento dello 0,5%.

La stessa analisi è stata fatta per ACTR web ed anche per questa applicazione risulta un incremento dello 0,5% di descrizioni con punteggio uguale a 10.

Appendice A

Il file di progetto è costituito dalle seguenti sezioni:

batch
INPUTFILE=nome e path assoluto del file di input che verrà codificato nella codifica batch
input
TEXTCOL=nome della colonna, del file di input, contenente il testo da codificare
FILTERCOL=nome della colonna, del file di input, avente la funzione di filtro
NCOLINPUT=numero di colonne del file di input da riportare nei file di output di CIRCE (sono sequenziali partendo da sinistra)
SEPCHAR=carattere asci di separatore di colonna del file di input
output
OSEPCHAR=carattere asci di separatore di colonna dei files di output
strategy
CSTRATNAME=nome della strategia
SCWIN= soglia massima
SCPOS= soglia minima
SCDIFF= differenziale (delta)
MAXMATCH=numero massimo di match (record), per ogni testo da codificare, che verranno scritti nei files di output
FILTERDROP=utilizzo o no del filtro
strategy
CSTRATNAME=….
SCWIN=….
SCPOS=…
SCDIFF=…
MAXMATCH=…
FILTERDROP=…
coding
DEFSTRAT=nome di una della precedenti strategie, definite all'interno del file, da utilizzare come default

Appendice B

Di seguito si riportano le possibili trasformazioni dei testi nel processo di standardizzazione (parsing):

Trasformazione	Fase/Descrizione	File dei dati
	Fase: Pre-trattamento	
Autotrim	caratteri iniziali e finali: toglie tutti i blank, tabulatori e newline	-
trimStrimmming	toglie tutti i blank, tabulatori e newline rispettivamente a sinistra e a destra dei caratteri indicati nei rispettivi file	trim_left.txt trim_right.txt
CCHR (*)	trasforma i caratteri indicati nella I col. in quelli della II col.	Cchr.txt
WCHR	trasforma i caratteri indicati nella I col. in quelli della II col.	WChr.txt
	Fase: Trattamento delle stringhe	
DCLS	cancella tutto ciò che è compreso tra incisi (tra il carattere/stringa della I col. e quello della II col.)	DCls.txt
DSTR	cancella le stringhe ritenute inutili	DStr.txt
RSTR	sostituzione di stringhe indicate nella I col. in quelle della II col.	rstr.txt
SepWord	trasforma in blank tutti i caratteri che non fanno parte della II colonna del file. Toglie tutti i blank all'inizio e alla fine e i doppi blank fra le parole, in modo che un solo blank separi ogni parola	WChr.txt
	Fase: Trattamento delle parole	
RWRD	consente la gestione dei sinonimi, degli errori di ortografia e delle parole ininfluenti. Sostituisce le parole indicate nella I col. con quelle della II e della col III (se esiste)	rwrd.txt
DWRD	consente la gestione di sinonimi a livello di coppie di parole. Sostituisce le coppie di parole indicate nelle col. I e II con quelle della III e della IV col. (se esiste)	dwrd.txt
HWRD	sostituisce le coppie di parole contenenti il trattino indicate nella I col. con quelle della II, della III e della IV col. (se esiste)	HWrd.txt
IWRD	cancella tutte le parole che contengono i caratteri indicati nel file	IWrd.txt
EXCP	contiene le parole del file non devono subire le successive trasformazioni (PRFX, SUFX e MCHR)	Excp.txt
PRFX	suffissi da eliminare	Sufx.txt
SUFX	prefissi da eliminare	Prfx.txt
MCHR	caratteri doppi o tripli da rimuovere	MChr.txt
	Fase: Post-elaborazione	
RDUP	cancella tutte le parole duplicate	
SORT	ordina le parole secondo un ordine alfabetico ascendente	

(*) Aggiunta rispetto ad ACTR v3 per evitare che i caratteri specificati subiscano un'ulteriore trasformazione nel successivo passaggio SepWord.

Appendice C

L'appendice descrive il software di supporto al pacchetto CIRCE per la verifica della coerenza dei file di *parsing*

Figura 1 - Utility per il controllo dei file di *parsing*

Il pacchetto mostrato in figura permette di realizzare dei controlli su alcuni file utilizzati dal *parsing*. Questo prodotto era già stato sviluppato per ACTR v3, ma è stato riscritto per adattarlo al nuovo sistema. Al momento non è richiamabile dalla GUI di CIRCE perché per il suo utilizzo è richiesto un certo livello di formazione/competenza.

La descrizione che segue, relativa ai controlli sui file di *parsing,* è stata estratta dal volume Istat *"Metodi e norme – n.41 2009"* a cui si rimanda per ulteriori dettagli informativi.

Il pacchetto CIRCE, al momento della creazione del contesto effettua solo un controllo finalizzato a non caricare i record "non validi", ossia quelli che, a seguito del *parsing* presentano la stessa descrizione (standardizzata). In questo caso si interrompe la creazione del contesto e viene creato il file dei duplicati, da analizzare successivamente per correggere i file di *parsing*.

Non viene effettuato alcun controllo di coerenza sui file di *parsing;* tale attività resta pertanto completamente a carico dell'utente. Poiché si è ritenuto che la gestione di questi controlli sia piuttosto pesante, considerando sia la complessità della classificazione che le dimensioni dei file dei sinonimi, si è provveduto a convertire il precedente software di supporto per ACTR, sviluppato per evidenziare eventuali incoerenze, allo scopo di lasciare la risoluzione al gestore dell'applicazione. I file sottomessi ai controlli sono:

- RSTR Replacement String;
- RWRD Replacement Word;
- DWRD Double Word.

I controlli sono effettuati internamente a ciascun file (un file su se stesso) e tra un file e l'altro,

mantenendo come ipotesi di partenza uno *strategy file* che esegua i processi di *parsing* secondo l'ordine RSTR => RWRD => DWRD. Il software è dotato di un'apposita interfaccia che consente l'elaborazione di tre fasi di controlli.

La **prima fase,** che viene lanciata cliccando sull'apposito bottone "FASE1", effettua una serie di controlli singolarmente su ciascun file; i passaggi realizzati sono i seguenti:

1. ordinamento del file, secondo i seguenti criteri:
   RSTR => ordinamento alfabetico sulla seconda colonna
   RWRD => ordinamento alfabetico sulla seconda colonna, quindi sulla terza
   DWRD => ordinamento alfabetico sulla terza colonna, quindi sulla quarta
2. eliminazione delle righe duplicate
3. eliminazione delle righe nelle quali i termini delle colonne di destra sono uguali a quelli delle colonne di sinistra (trasformazione A => A).

La **seconda fase**, invece, è finalizzata a mettere in luce diversi tipi di incoerenze all'interno di ciascuno dei file citati ed a memorizzarle su appositi file Excel. In particolare si procede:

1. all'individuazione delle trasformazioni incoerenti, quali ad esempio:
   trasformazione di A => B
   trasformazione di A => C
2. all'individuazione delle trasformazioni ricorsive, quali ad esempio:
   trasformazione di A => B
   trasformazione di B => C
3. oppure
   trasformazione di A => Z B
   trasformazione di B => C D

Come può vedersi dalla Figura 1, viene richiesta l'esecuzione della procedura, relativa ai tre file di *parsing* citati, in una sequenza prestabilita; le incoerenze sono registrate in appositi file Excel. Al termine di ciascun passaggio, non soltanto è evidenziato il bottone "VISUALIZZA", che consente di editare i file Excel, ma sono valorizzati una serie di contatori, nei quali sono riportati i totali delle righe da correggere.

Con la **terza fase**, infine, vengono realizzati i controlli tra il file delle RWRD e quello delle DWRD.

In particolare si procede:

1. all'individuazione di coincidenze tra la prima colonna del file RWRD e le prime due colonne del file DWRD, ad esempio:
   trasformazione in RWRD di A => Z
   quindi trasformazione in DWRD di A B=> C D
   oppure trasformazione in DWRD di B A => C D
2. all'individuazione di coincidenze tra la prima colonna del file RWRD e le seconde due colonne del file DWRD, quali ad esempio:
   trasformazione in RWRD di A => B
   quindi trasformazione in DWRD di C D => A B
   oppure trasformazione in DWRD di C D => B A

Si fa presente, comunque, che quest'ultimo tipo di coincidenze non costituisce necessariamente un errore, quindi non è indispensabile un intervento correttivo, può essere sufficiente verificare che non si tratti di una svista, ma di una trasformazione voluta.

L'utilizzo corretto di questa applicazione implica che, ogni qual volta si effettui un intervento correttivo in un qualsiasi passaggio di ciascuna fase, sia opportuno eseguire nuovamente la procedura dall'inizio, in modo da evidenziare le effettive righe incoerenti corrispondenti all'ultima versione di ciascun file di *parsing.*