ALMA MATER STUDIORUM ● UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

DIFFER - ELABORAZIONE E RAPPRESENTAZIONE DI MODIFICHE A DOCUMENTI STRUTTURATI

Relatore: Chiar.mo
Prof. Vitali Fabio

Presentata da:
Dondi Simone

Sessione Unica

Anno Accademico 2017-2018

Indice:

Differ - elaborazione e rappresentazione di modifiche a documenti strutturati

1. Introduzione

Scopo di questa dissertazione è presentare una modalità di visualizzazione delle modifiche ad un documento indicativa del livello di priorità e del tipo di edit, sfruttando un raggruppamento di quelle che sono operazioni puramente meccaniche in operazioni di più alto livello, strutturali ed eventualmente semantiche.
I principali editor usati e i principali sistemi di versionamento dei documenti offrono delle modalità di visualizzazione delle differenze scarne di significato, rendendo difficile per l'utente l'attribuzione di senso ai cambiamenti evidenziati sul testo.
Ho quindi cercato di porre un rimedio al problema fornendo una struttura di edit e alcuni schemi visivi capaci di conferire una intuitiva attribuzione di significato alle modifiche.
Come esempio a dimostrazione del problema e della possibile soluzione, userò una delle modifiche che ho elaborato all'interno del mio progetto.

Vecchio testo:	*Nuovo testo:*
che consente al titolare tramite la chiave privata e al destinatario tramite la chiave pubblica	che consente al titolare tramite la chiave privata e a un soggetto terzo tramite la chiave pubblica

Così visualizzata, la modifica è di difficile percezione. Si tratta dell'eliminazione di "al destinatario" e dell'inserimento, al suo posto, di "a un soggetto terzo". Se vogliamo rappresentare attraverso la struttura dati JSON le precedenti operazioni, ad un livello meccanico questo potrebbe essere il risultato:

```
{
      id: "EDIT",
      operation: "DEL",
      position: 8762,
      content: "al destinatario"
}, {
      id: "EDIT",
      operation: "INS",
      position: 8762,
      content: "a un soggetto terzo"
}
```

È evidente come questa rappresentazione sia di per sé concettualmente corretta, ma che non fornisca le necessarie informazioni per una comprensione semantica adeguata di ciò che l'autore della modifica intendesse effettuare.
Se invece aggiungiamo una tipizzazione di ciò che è successo, raggruppando le due operazioni meccaniche in una operazione strutturale, la modifica assume un senso e riceve un'immediata attribuzione di significato:

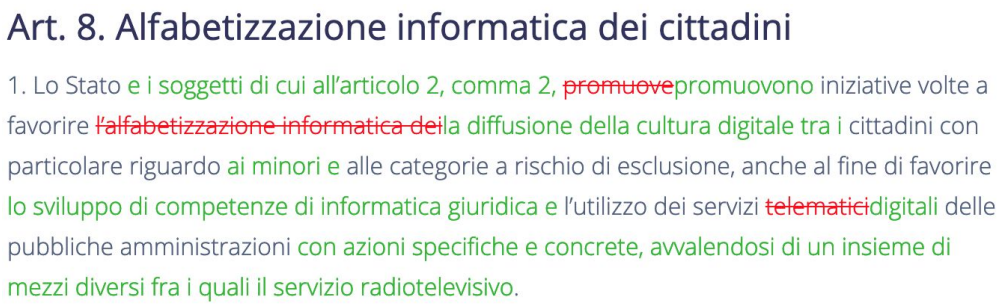
```
 {
    id: "STRUCTURAL",
    operation: "TEXTREPLACE",
    items: [{
        id: "EDIT",
        operation: "DEL",
        position: 8672,
        content: "al destinatario"
}, {
        id: "EDIT",
        operation: "INS",
        position: 8672,
        content: "a un soggetto terzo"
    }]
}
```

Infatti, la variazione in questione consiste in una sostituzione di una stringa, un insieme di parole, con un'altra, nella stessa posizione.

Sono molti gli scenari nei quali una rappresentazione accurata e significativa delle modifiche ad un documento suscita interesse e fornisce utilità. Ad esempio, negli scenari di editing collaborativo, di elaborazione e stesura di un disegno di legge, di piattaforme per la gestione del processo di development di software, di traduzione di testi, e tanti altri. Esistono già molti tool, editor e algoritmi che si occupano di quello che in gergo è chiamato diff-ing, cioè operazioni di rilevazione, memorizzazione e visualizzazione delle modifiche a documenti di testo. Però vi sono contemporaneamente molte lacune nella maggior parte di essi, soprattutto per quel che riguarda la caratterizzazione e la categorizzazione delle variazioni.

Quindi quello che ho cercato di fare è stato proporre una soluzione a questa tipologia di problemi, sfruttando una struttura dati capace di offrire una prioritizzazione dei cambiamenti in un documento strutturato gerarchicamente, e sviluppando una metodologia di visualizzazione capace di offrire informazioni sulle modifiche, e quindi possibilità di personalizzazione per l'utente.

Ho quindi creato una libreria Javascript che, ricevendo in input un documento di testo e una lista di modifiche appositamente impostata, restituisca in output diversi stili di visualizzazione relativamente al documento modificato e alle modifiche stesse. Più nello specifico, partendo da una struttura dati contenente le modifiche ed il documento originale, esegue delle elaborazioni e offre alcuni servizi: il documento con le variazioni applicate caratterizzate attraverso una struttura descrittiva del tipo di modifica, una modalità comparativa delle operazioni effettuate e un raggruppamento temporale delle modifiche.



Esempio di possibile output

Questa libreria farà parte di un progetto per editing di documenti ingegneristici molto strutturati, composto da un motore di creazione di strutture dati delle edits a partire da due versioni di un documento, e dalla mia libreria di visualizzazione di esse, opportunamente modificata per poter interagire con il motore suddetto.

2. Motori di visualizzazione di edits

In alcuni scenari di editing collaborativo, come la scrittura di articoli accademici, il development di software o piattaforme wiki-based, i redattori trovano utilità nel tenere traccia delle modifiche ad un documento e delle sue varie versioni. Esistono vari tool che effettuano questa operazione, in gergo chiamata “diffing”. È bene chiarire la differenze fra piattaforma di change tracking o revision control e fra algoritmi di diffing e visualizzazione dei loro risultati.

Il revision control è, in Informatica, la gestione di versioni multiple di un documento, solitamente all’interno del contesto di editing, con annesso change tracking, cioè la memorizzazione di tutte le modifiche effettuate. Entrambi rendono possibile quindi anche una catalogazione temporale delle varie versioni e di tutto il processo di editing, anche di modifiche in seguito non accettate e scartate.

Gli algoritmi di diffing sono invece procedure mirate a generare delle liste di modifiche, cosiddette delta, a partire da due versioni di un documento.

In questa sezione ho effettuato un’analisi delle principali piattaforme che offrono questo tipi di servizi.

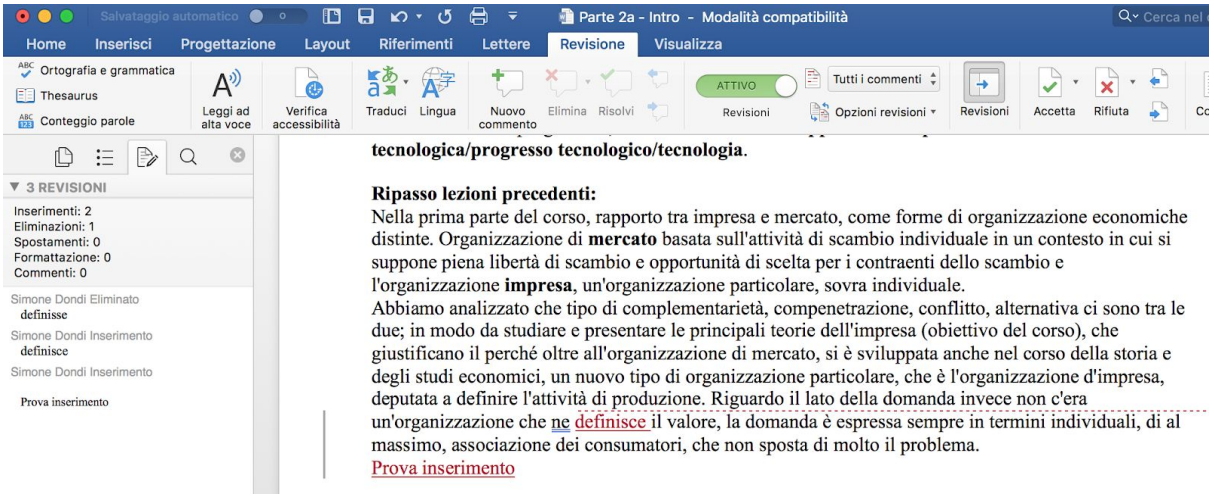
In particolare, per quel che riguarda gli editor di testo, ho analizzato i servizi di change tracking di Microsoft Word, Apache Open Office e Apple Pages. Per quanto concerne piattaforme web dotate di questo tipo di funzionalità, ho preso in considerazione Wikipedia e GitHub. Inoltre, ho riassunto alcuni articoli scientifici riguardanti critiche, approfondimenti ed estensioni a suddette funzionalità, soprattutto sulla cronologia delle versioni di Wikipedia.

Infine, ho effettuato uno studio su sei algoritmi di visualizzazione di modifiche e varianti, analizzandone i pregi e i difetti.

2.1 Editor di testo

Esistono in letteratura svariate tipologie di visualizzazione di diff e varie piattaforme che offrono servizi di change tracking e revision control. Il mio studio del contesto scientifico e tecnologico alla base di questo tipo di tecnologie è iniziato dai più famosi ed utilizzati editor di testo: Microsoft Word [MC19a], Apache Open Office [ASF18] e Apple Pages [AI19]. Questi word processors mettono a disposizione dell'utente degli strumenti di revisione delle modifiche, con similitudini e differenze. In tutti e tre i programmi questa funzione deve essere attivata e permette di tenere traccia degli inserimenti e delle cancellazioni effettuate sul documento man mano che lo si modifica.

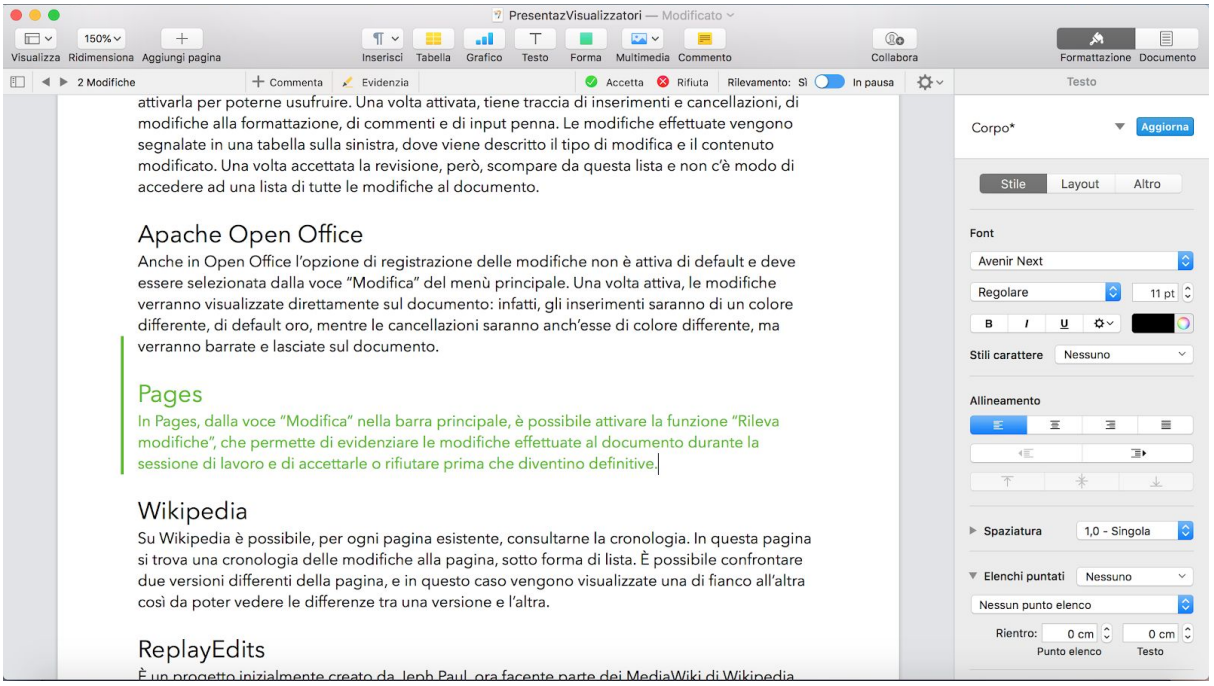
In Word, una volta attivata la funzione, le rimozioni vengono visualizzate barrate mentre gli inserimenti vengono sottolineati. Inoltre, nella parte sinistra dell'interfaccia grafica vi è una tabella dove vengono conteggiate le modifiche in base al tipo, in modo da avere una sorta di resoconto di esse.

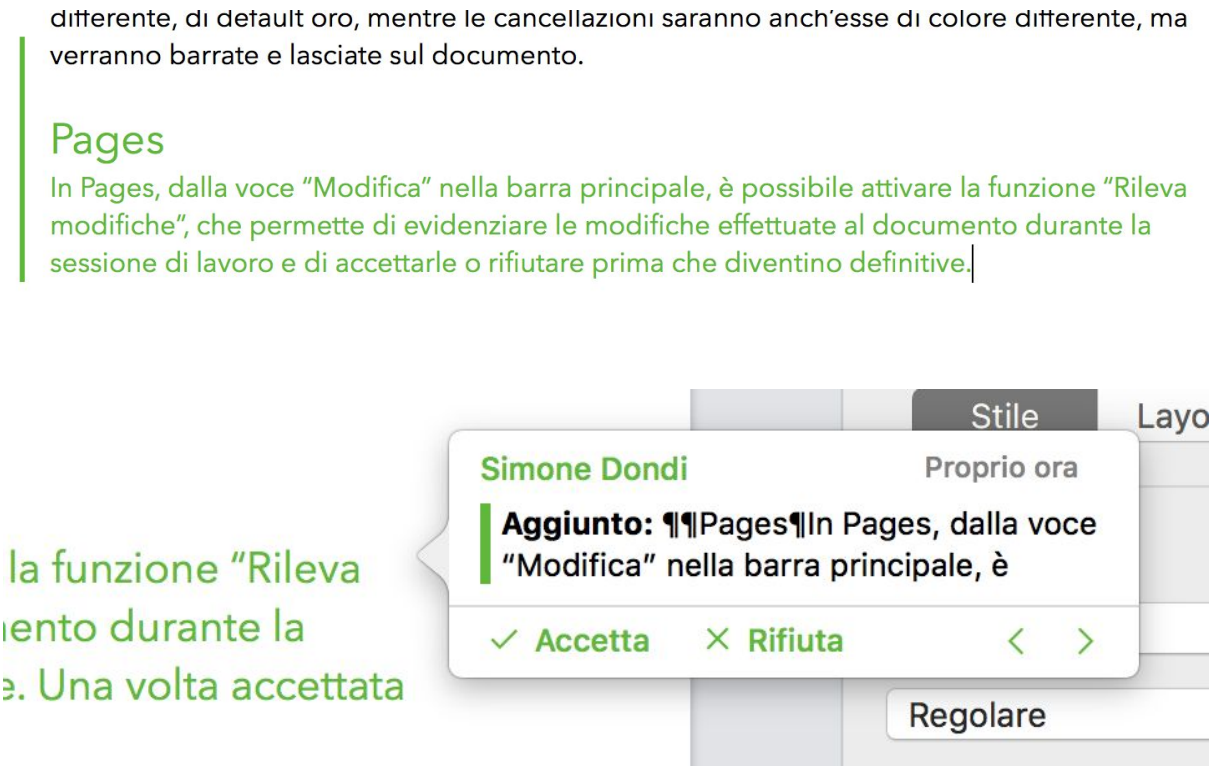


Controllo Revisioni su MS Word

In Open Office, invece, il comportamento è molto simile a Word, ma cambia leggermente il metodo di visualizzazione sul documento dei cambiamenti: infatti sia le cancellazioni che gli inserimenti vengono barrati e sottolineati, ma assumono anche una colorazione oro. Inoltre non è presente la tabella riassuntiva.

Anche in Pages il servizio è simile, differisce nel colore, che qui è verde; e non si dispone nuovamente della tabella riassuntiva.





Esempi di Rileva Modifiche in Apple Pages

Un elemento comune di questi editor è che le modifiche non vengono salvate e non vengono memorizzate varie versioni del documento. Infatti, una volta accettate le modifiche effettuate e una volta rese definitive, non vi è modo di mantenerne una traccia. Semplicemente scompaiono diventando parte del documento. Ovviamente non è il comportamento in sé ad essere inadeguato, è naturale che se una modifica è ritenuta corretta debba diventare definitiva; però non c'è modo di mantenere traccia della storia del documento.

Riguardo questo punto, Coakley et al. [CMT14] propongono un plugin per Microsoft Word che permette appunto di salvare versioni di un documento, che loro chiamano revisioni. Attraverso una struttura XML, ogni volta che un utente effettua cambiamenti al documento e li salva, il loro plugin salva una nuova revisione per quel documento e la aggiunge alla storia di revisioni del documento. Purtroppo questo approccio ha delle limitazioni dovute al fatto di essere un plugin per Microsoft Word, che significa non poter usufruire di questa tecnologia all'infuori dell'editor di testo in questione.

Figure 4: A screenshot of Microsoft Office with the version-aware add-in loaded, showing the graph visualization.

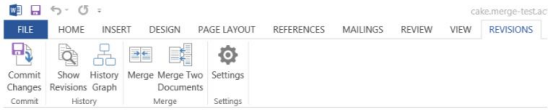


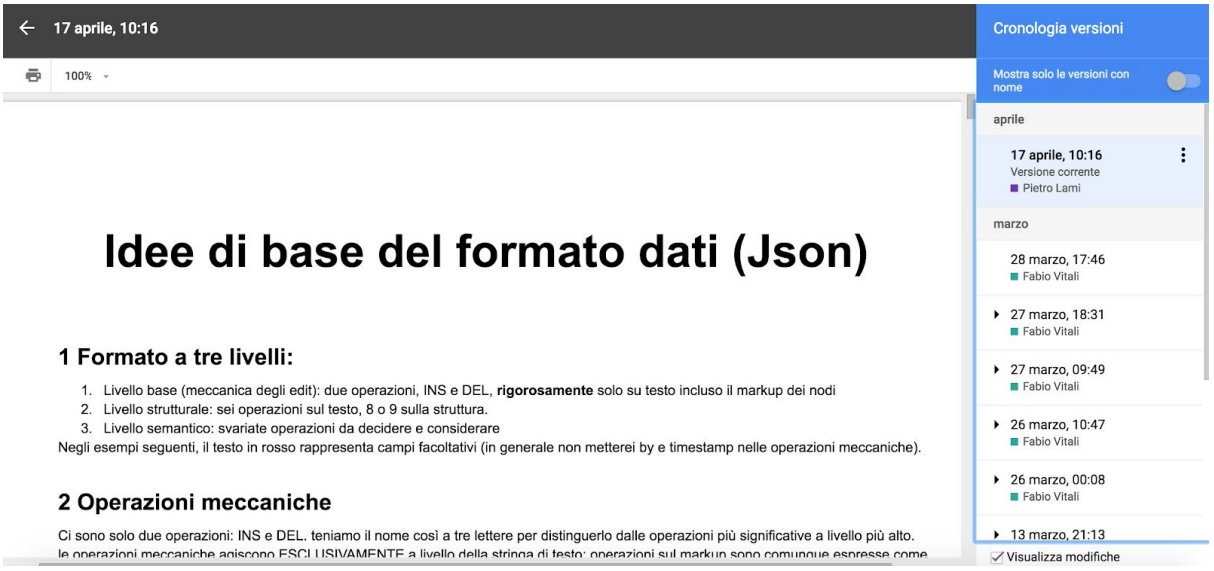
Figure 5: The Microsoft Office ribbon tab that presents controls for managing version-aware Word documents.

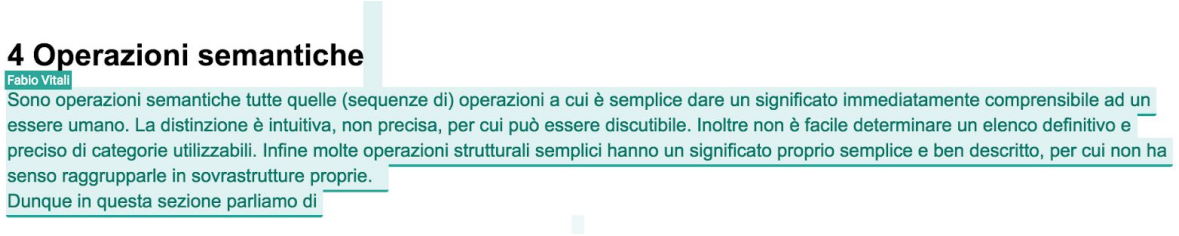
```
<?xml version="1.0" encoding="utf-8"?>
<revision-history
  xmlns="http://www.cs.uwm.edu/molhado/revision-
    history"
  current="24f50581-38ba-46ae-839d-14d8b8f4e0f4">
<revision id="e37bbc3f-e056-4259-a081-5ae7f2c8153e"
  author="john@domain"
  timestamp="2013-11-23T09:36:49"
  location="/history/e37bbc3f-e056-4259-a081-5
    ae7f2c8153e">
</revision>
<revision id="24f50581-38ba-46ae-839d-14d8b8f4e0f4"
  author="john@domain"
  timestamp="2013-11-23T09:38:46"
  location="/history/24f50581-38ba-46ae-839d-14
    d8b8f4e0f4">
  <parent id="e37bbc3f-e056-4259-a081-5ae7f2c8153e"/>
</revision>
</revision-history>
```

Figura di [CMT14]

Un'altro editor di testi online che offre un servizio di change tracking e revision control è Google Docs [GO19]. Google Docs permette di creare, modificare e salvare documenti all'interno della piattaforma, offrendo anche la possibilità di lavorare

contemporaneamente in più persone sullo stesso documento. Il servizio tiene traccia delle modifiche effettuate e dell'utente che le ha eseguite, ed è possibile accedere alla cronologia delle revisioni per avere un quadro completo della storia del documento. Le limitazioni di questo servizio, però, consistono nel fatto che in Google Docs è possibile solamente lavorare su determinati formati, e non vi è possibilità di mantenere un registro delle modifiche effettuate una volta esportato il documento al di fuori della piattaforma.





Cronologia versioni in Google Docs

2.2 Piattaforme web

Continuando lo studio sul contesto delle tecnologie di diffing, ho analizzato due siti web che offrono all'interno della loro piattaforma la possibilità di visionare differenze tra documenti: Wikipedia [WF19] e GitHub [MC19b].

Wikipedia è un'enciclopedia libera online che si propone di raccogliere quante più informazioni possibili in ogni campo e argomento. Per ogni pagina è possibile accedere alla sua cronologia di modifiche ed effettuare una comparazione tra due diverse versioni di essa, visualizzando le differenze tra una e l'altra. In questo caso, le limitazioni risiedono nel fatto che non è semplice capire cosa è stato modificato, che tipo di modifica è stata eseguita e per quale motivazione, nonostante l'interfaccia grafica sia gradevole e tutto sommato chiara. Questo perché spesso viene visualizzato anche il markup proprio della piattaforma, di per sé poco comprensibile per un non addetto ai lavori, e perché sono poche le informazioni sul tipo di cambiamento eseguito.

Esempio di Cronologia delle modifiche della pagina Wikipedia "Stretto di Mackinac"

Inoltre, anche la storia delle versioni di una pagina può intrinsecamente contenere delle ambiguità. Infatti, a causa della natura dei cosiddetti reviewers, cioè coloro che possono modificare una pagina Wikipedia, non sempre le modifiche hanno un senso

compiuto o un fondo di verità, e spesso ci si trova in situazioni dove i cambiamenti sono stati eliminati e si è tornati alla precedente versione della pagina.

Viegas, Wattenberg e Dave [VWD04] hanno affrontato uno studio sulla visualizzazione delle versioni delle pagine Wikipedia. La loro proposta è un'interfaccia grafica per la visualizzazione basata su linee rappresentative per ogni versione, collegate tra loro: ogni versione è rappresentata da una linea in cui ogni segmento corrisponde alle modifiche effettuate da un autore, contrassegnato da un particolare colore, con una lunghezza proporzionale alla quantità di cambiamenti. Ogni segmento di ogni versione è poi collegato alla versione successiva, indicando se e quanto del contributo precedente sia sopravvissuto nella versione successiva. Il risultato è una visualizzazione intelligente della storia di una pagina Wikipedia che permette ad un utente di analizzare immediatamente il percorso subito dalla pagina e di avere informazioni utili riguardanti le modifiche e gli autori di esse. Ma la soluzione proposta è circoscritta alla piattaforma Wikipedia, nonostante offra spunti di riflessione molto importanti sulle possibili modalità di visualizzazione degli storici delle versioni di un documento in generale.

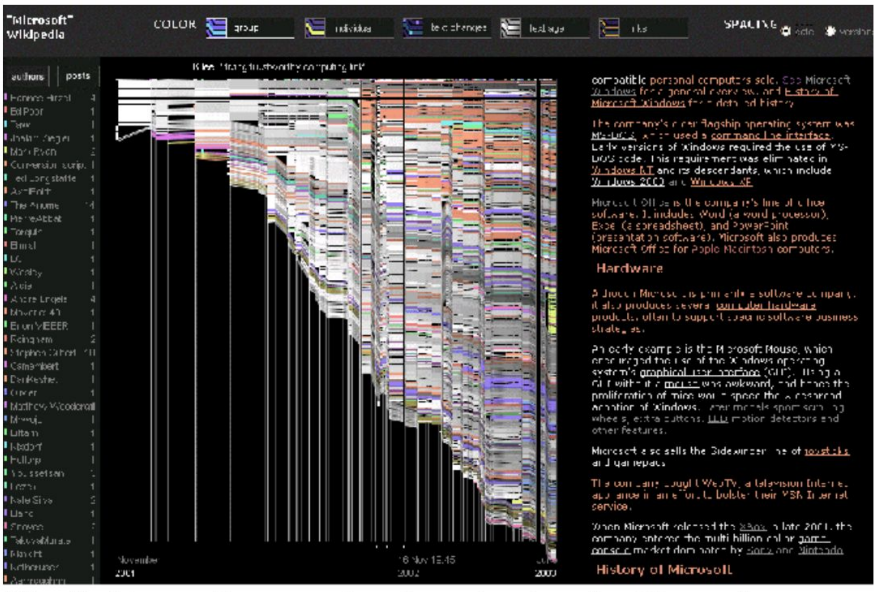


Fig 3: *history flow user interface showing the Microsoft page on*

Fig 3: history flow user interface showing the Microsoft page on Wikipedia; on the right we see the contents of the page, on the left we see all the authors who have contributed to this page; the center panel shows the visualization

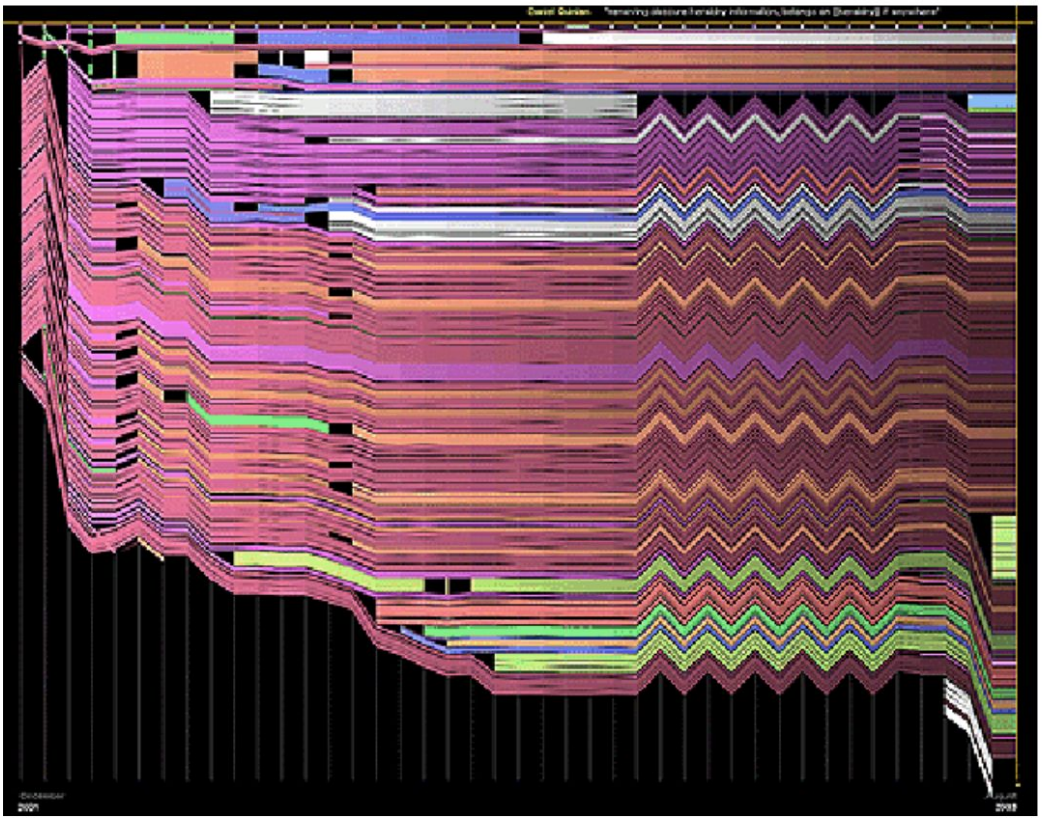


Fig 6: “Chocolate” page spaced out by number of versions; we can see the zigzag pattern of an edit war.

Figure di [VWD04]

Sabel [SA07] invece riguardo a ciò propone una struttura di rappresentazione della storia di una pagina ad albero e non cronologicamente lineare, assegnando ad ogni versione un coefficiente che funge da peso specifico della versione, e collegando versioni tra loro attraverso relazioni padre-figlio. In questo modo viene creata una struttura per le versioni più rappresentativa e più utile per comprendere l’andamento storico dell’editing della pagina, in quanto le varie versioni non sono più semplicemente elencate in modo cronologico ma assumono un significato umanamente riconoscibile.

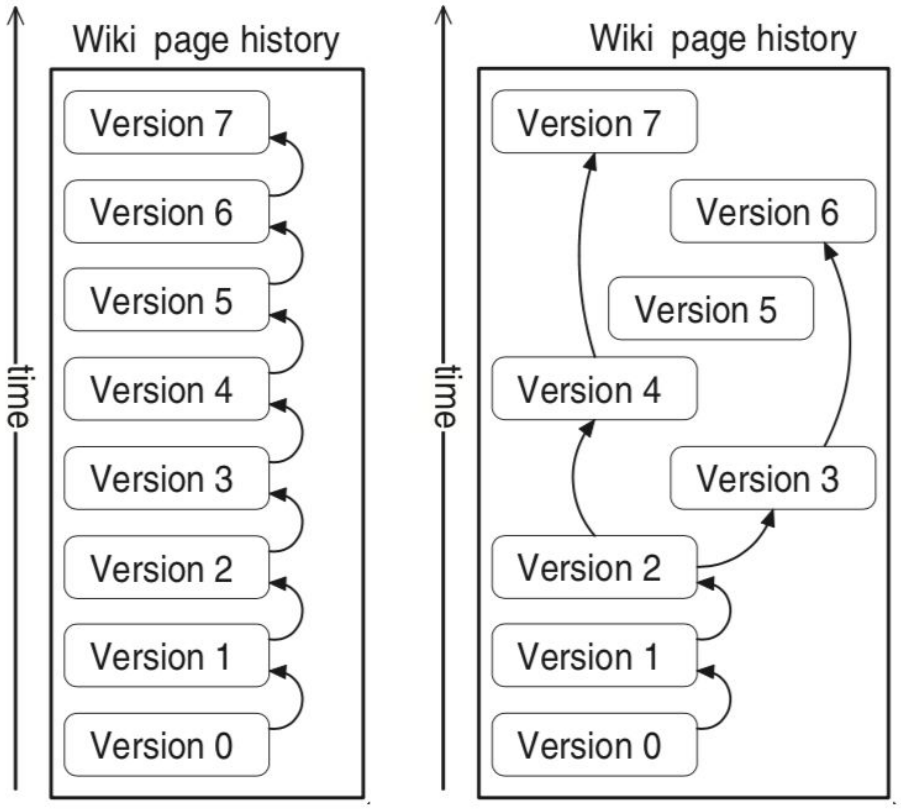


Figure 1. Linear (left) and tree (right) representations of wiki page history

Figura di [SA07]

Partendo dalle considerazioni dei due lavori precedentemente illustrati, Ektrand e Riedl [ER09] hanno sviluppato una visualizzazione alternativa e più accurata della history view di Wikipedia. Infatti, prendendo spunto soprattutto dal lavoro di Sabel, hanno considerato ogni revisione in relazione al tipo di modifica che ha apportato, concentrandosi su quelle di tipo revert, cioè quelle revisioni che non hanno fatto altro che riportare lo stato della pagina ad uno stato precedente, rifiutando di fatto le modifiche fatte successivamente. Hanno creato quindi una struttura ad albero della storia di una pagina, con la particolarità che se una revisione è di tipo revert, viene posta come figlia della revisione precedente a cui si è ritornati. Una volta creata la struttura, essa viene visualizzata direttamente a fianco della classica visualizzazione della storia di una pagina, indicando con l'aiuto di pallini colorati, linee e frecce di che tipo di revisione si tratta e a quale versione precedente fa riferimento.

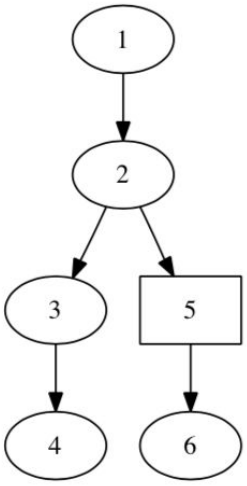


Figure 2: Tree built from revisions in "Chocolate".

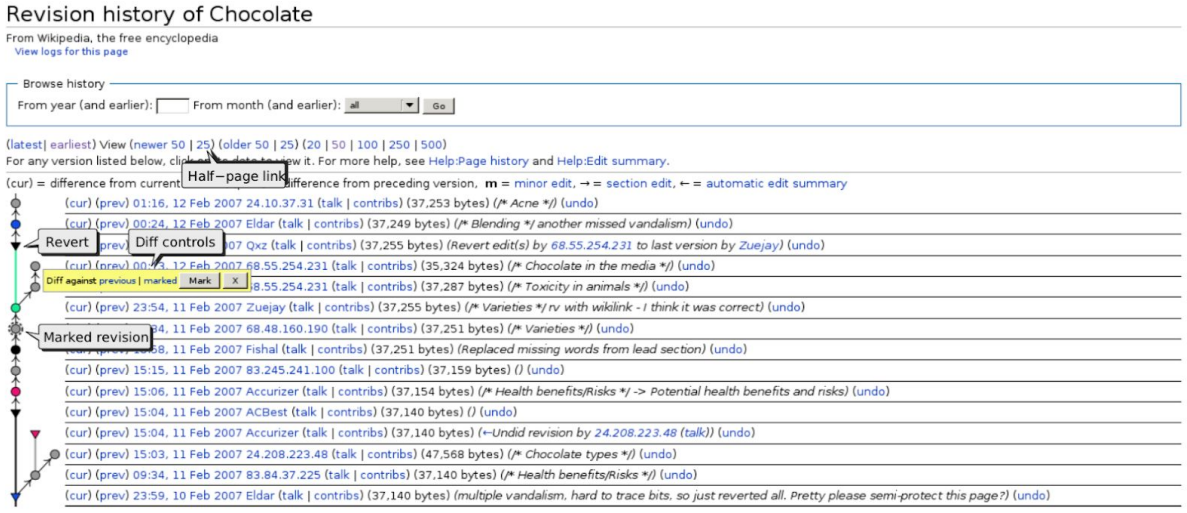


Figure 3: History view with a tree visualization. The top revert is revision 5 in Figure 2.

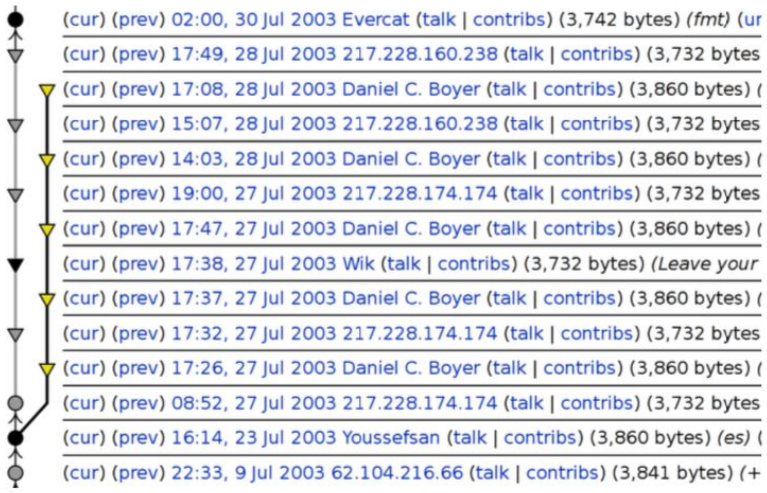


Figure 5: Revert war early in the history of "Chocolate".

Figure di [ER09]

Altri ricercatori che hanno effettuato uno studio a questo proposito sono Fong e Biuk-Aghai [FBA10]. Essi, partendo dallo studio delle cosiddette Wiki, in particolare Wikipedia, hanno dedotto che la struttura usata e la visualizzazione della storia dei documenti non forniva informazioni utili e presentava difficoltà di comprensione relativamente alla tipologia di modifica effettuate ad ogni versione. Le modifiche vengono presentate semplicemente come "differenze di parole", e non c'è nessuna analisi di alto livello su di esse. Hanno quindi immaginato e tentato di realizzare un software di analisi della storia delle modifiche che, partendo dalle modifiche meccaniche di basso livello, ne facesse un'analisi ed astraesse alcune informazioni relative al significato di esse, fornendo come output un sommario storico del documento.

Figure 3. Edit analysis tab extension in MediaWiki

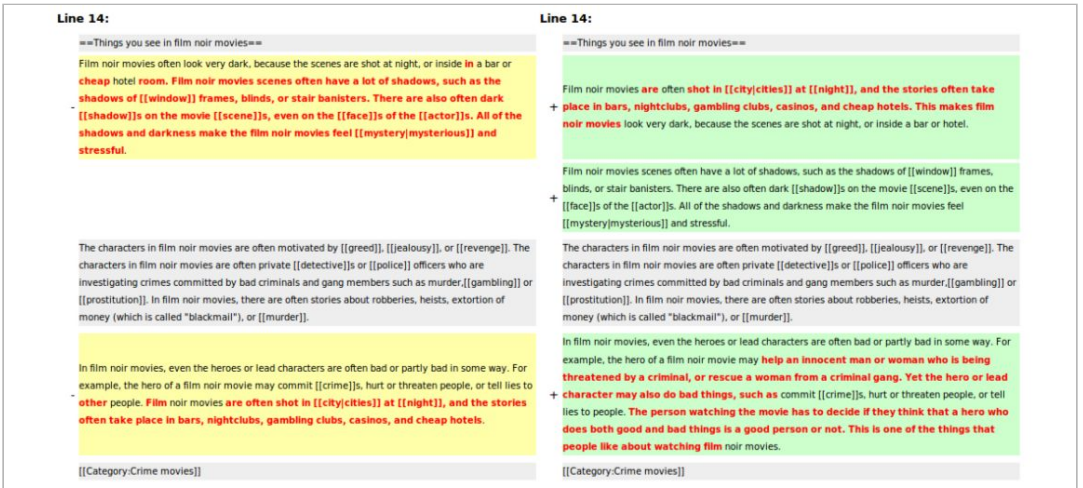


Figure 4. Difference page of article "Film noir" produced by MediaWiki analyzer (background colours: yellow/green – old/new version paragraph changed, gray – paragraph unchanged; text colour: red – word changed (deleted, inserted))

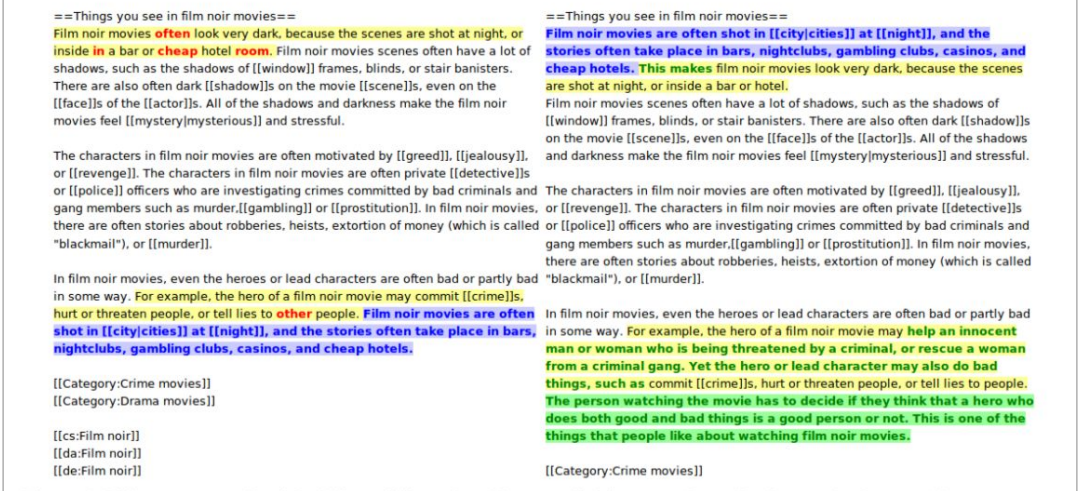


Figure 5. Difference page of article "Film noir" produced by our edit history analyzer (background colours: yellow – sentence changed, blue – sentence moved, green – sentence added; text colours: red – word deleted; blue – word moved, green – word inserted)

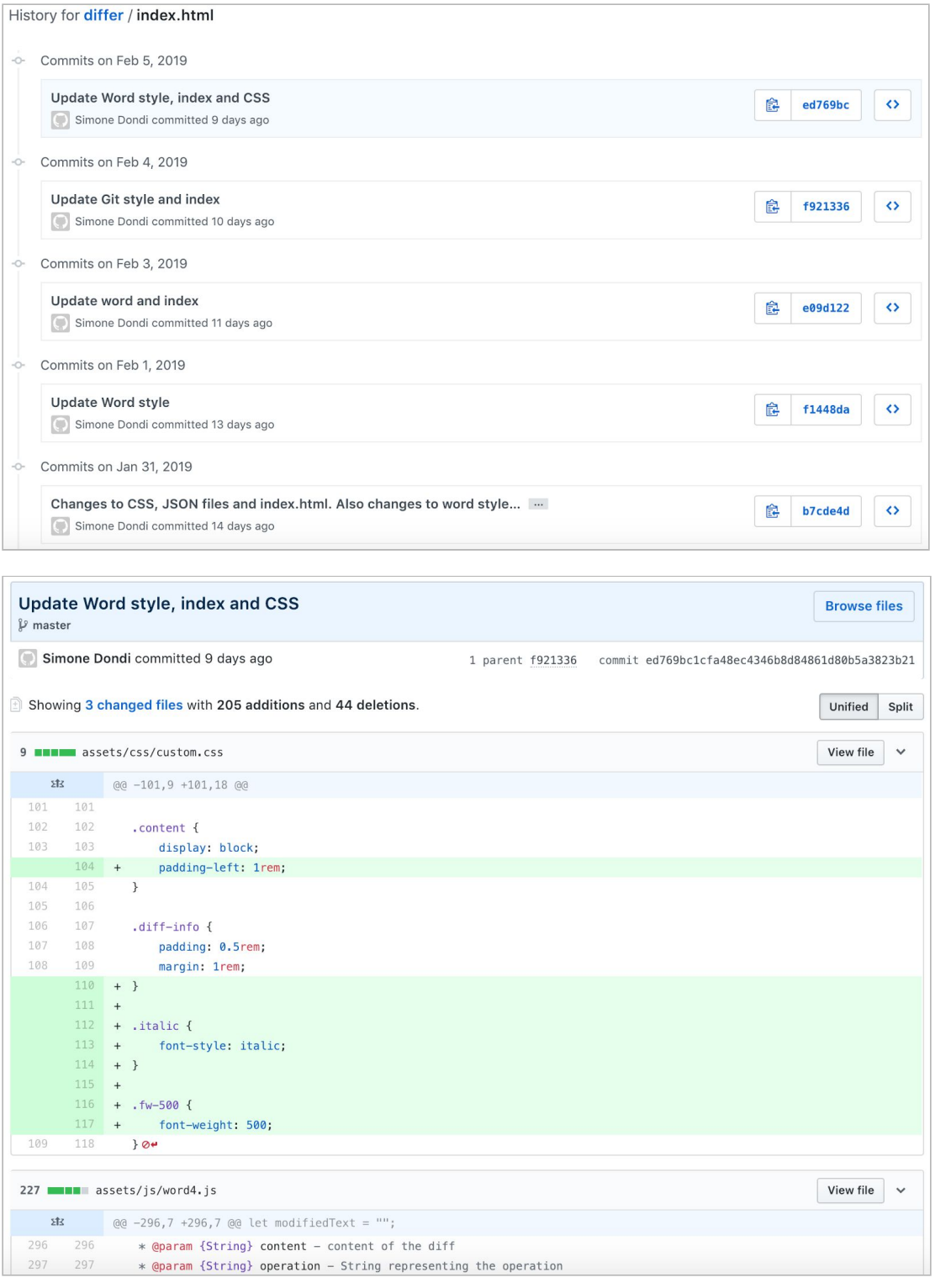
Figure di [FBA10]

Sia per quel che riguarda il lavoro di Sabel che per quello di Fong et al., il punto di partenza del loro studio è molto interessante e si muove nella giusta direzione di creare una visualizzazione arricchita e chiara, però è limitato a migliorare piattaforme che dispongono già di un servizio di history revision e la loro soluzione è altrettanto limitata a queste piattaforme, non permettendo di usufruire dei loro risultati in qualsiasi altra situazione. Stesso discorso per il lavoro di Ektrand e Riedl, il quale inoltre si focalizza esclusivamente sulla piattaforma Wikipedia.

Per quel che riguarda GitHub, il discorso è molto simile a Wikipedia. GitHub è una piattaforma di repository per progetti software e anche qui è possibile visionare la storia di un documento, in particolare di qualsiasi tipo di documento che forma la repository, quindi file di codice, di testo, di configurazione, ecc. Una volta selezionato un file, è possibile accedere alla "History", dove le modifiche vengono raggruppate temporalmente e in base a commit (operazioni tecniche che eseguono l'upload di nuove versioni di uno o più documenti contemporaneamente all'interno della repository). Selezionando un particolare commit, vengono mostrati i file sui quali sono stati effettuati cambiamenti e le modifiche stesse, riga per riga, evidenziando in rosso una riga dove è stata effettuata una rimozione e in verde una in cui è stato effettuato un inserimento. Inoltre è possibile visualizzare le modifiche in modalità diverse: in modo unificato oppure in una modalità comparativa, simile a Wikipedia; solo le righe che hanno subito cambiamenti oppure l'intero file.

A proposito di questa piattaforma si potrebbe fare una digressione sulle modalità di controllo delle versioni durante il processo di development di un software o delle funzionalità a supporto della programmazione in team di un singolo progetto, ma questo tipo di considerazioni si allontanerebbero da quello che è l'argomento che voglio trattare. Prenderò quindi in considerazione solo l'aspetto relativo alla visualizzazione grafica delle modifiche ad un documento, anche se in questo caso si tratta di file di codice, che da un punto di vista strutturale sono ben diversi da dei documenti di testo, specialmente se questi ultimi sono descritti attraverso un linguaggio strutturale di markup. Quindi riguardo alla visualizzazione di modifiche, GitHub evidenzia l'intera riga in cui è avvenuto un cambiamento, e all'interno della riga stessa, evidenzia in modo più marcato i caratteri o la stringa che è stata modificata. In questo modo isola efficacemente le righe di codice che hanno subito

una variazione, ma rende meno intuitivo capire esattamente cosa è stato modificato, perché per farlo è necessario osservare sia le righe in cui è avvenuta una rimozione sia quelle in cui è avvenuto un inserimento.

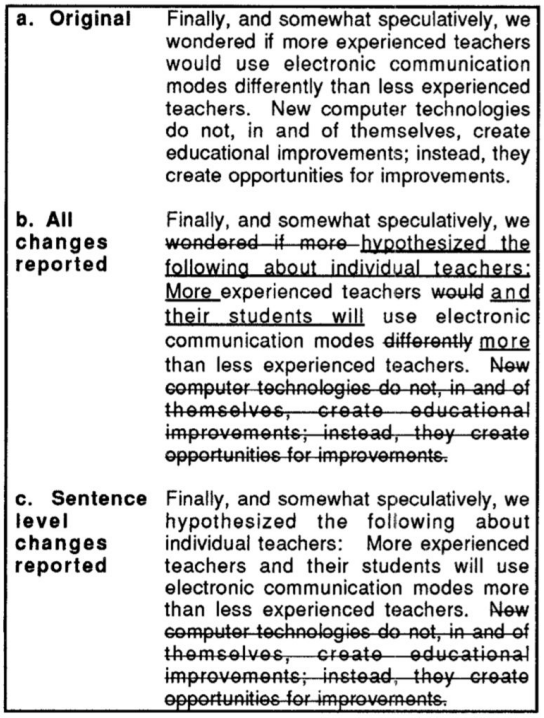


Esempi di History di GitHub

2.3 Visualizzazioni di algoritmi di diffing

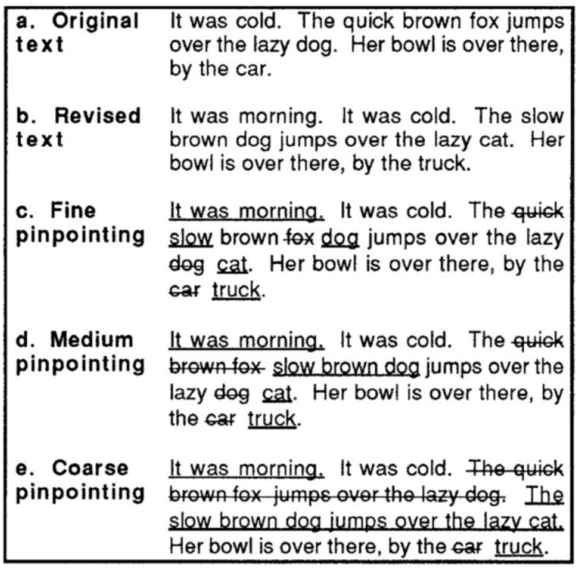
Passando allo studio di algoritmi di diffing che offrono, oltre al riconoscimento delle modifiche ad un documento, anche la loro visualizzazione grafica, ne ho trovati alcuni che offrono spunti di riflessione importanti.

Già nel 1992 Neuwirth e colleghi [NCK92] si chiedevano quali fossero i bisogni e le necessità da parte di vari autori nella visualizzazione delle modifiche ad un documento, specialmente se la scrittura e la revisione di esso era affidata a più persone. Essi effettuarono riflessioni inerenti a diverse questioni, quali la quantità e il tipo di cambiamenti che devono essere mostrate, i bisogni di differenti autori in base al tempo e al ruolo. Hanno quindi studiato diversi tipi di rappresentazione derivanti appunto dalle diverse necessità. Ad esempio, vi possono essere situazioni dove non tutte le modifiche devono essere visualizzate mentre altre dove questa funzionalità è necessaria. Oppure vi possono essere fruitori che preferiscono un livello di specificità differente, quindi ad esempio nella sostituzione di una parola con un'altra non tenere conto se un carattere rimane invariato e mostrare l'intera parola cancellata e l'intera parola inserita ("dog" con "fox", dove ad un livello di carattere vi sono due sostituzioni di lettere mentre ad un livello di parola vi è una sostituzione dell'intera parola).



a. Original	Finally, and somewhat speculatively, we wondered if more experienced teachers would use electronic communication modes differently than less experienced teachers. New computer technologies do not, in and of themselves, create educational improvements; instead, they create opportunities for improvements.
b. All changes reported	Finally, and somewhat speculatively, we ~~wondered if more~~ hypothesized the following about individual teachers: More experienced teachers ~~would~~ and their students will use electronic communication modes ~~differently~~ more than less experienced teachers. ~~New computer technologies do not, in and of themselves, create educational improvements; instead, they create opportunities for improvements.~~
c. Sentence level changes reported	Finally, and somewhat speculatively, we hypothesized the following about individual teachers: More experienced teachers and their students will use electronic communication modes more than less experienced teachers. ~~New computer technologies do not, in and of themselves, create educational improvements; instead, they create opportunities for improvements.~~

Figure 2. All changes reported vs. sentence level changes.



a. Original text	It was cold. The quick brown fox jumps over the lazy dog. Her bowl is over there, by the car.
b. Revised text	It was morning. It was cold. The slow brown dog jumps over the lazy cat. Her bowl is over there, by the truck.
c. Fine pinpointing	It was morning. It was cold. The ~~quick~~ slow brown ~~fox~~ dog jumps over the lazy ~~dog~~ cat. Her bowl is over there, by the ~~car~~ truck.
d. Medium pinpointing	It was morning. It was cold. The ~~quick brown fox~~ slow brown dog jumps over the lazy ~~dog~~ cat. Her bowl is over there, by the ~~car~~ truck.
e. Coarse pinpointing	It was morning. It was cold. ~~The quick brown fox jumps over the lazy dog.~~ The slow brown dog jumps over the lazy cat. Her bowl is over there, by the ~~car~~ truck.

Figure 3. Pinpointing changes.

Figure di [NCK92]

La tecnica Deep Diff [SQN10] prende questi concetti (i concetti di visualizzazione di diff) e cerca di arricchire la visualizzazione delle modifiche sfruttando le informazioni temporali derivanti dalle versioni del documento. A partire dall'ultima, mostra in essa le parti modificate nel tempo con colorazioni differenti a seconda della quantità e della "giovinezza" delle modifiche subite in quel punto.

Questo tipo di visualizzazione arricchisce la quantità di informazioni immediatamente fruibili all'occhio umano, ma solo in termini temporali e quantitativi, senza rilevare anche le tipologie e le motivazioni alla base dei cambiamenti.

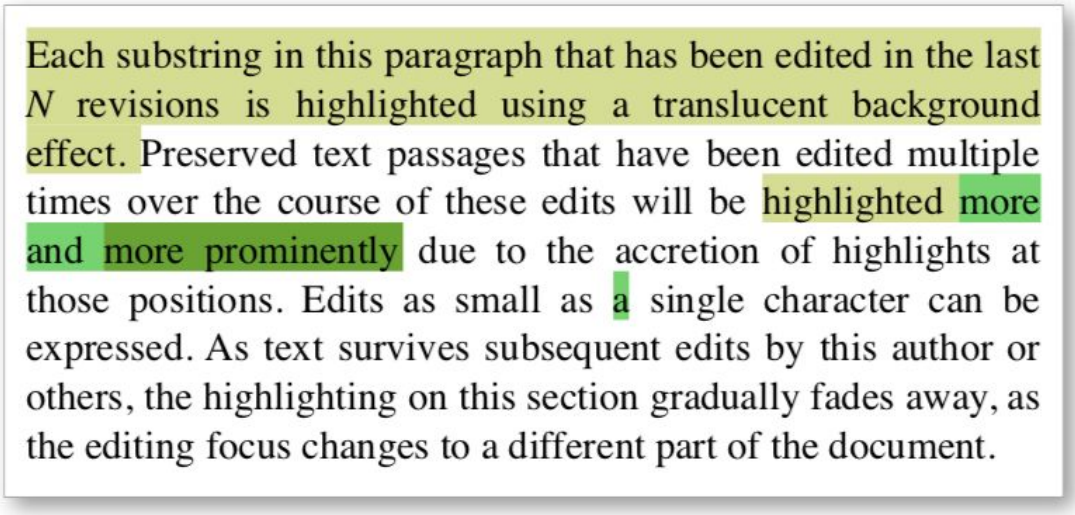


Figure 1: A segment of text that has been processed with the Deep Diffs technique. Highlighted green areas have been edited recently. As a passage is edited more frequently by one or many editors, the highlighting strengthens in that area so that other author's attention is called to this part of the text.

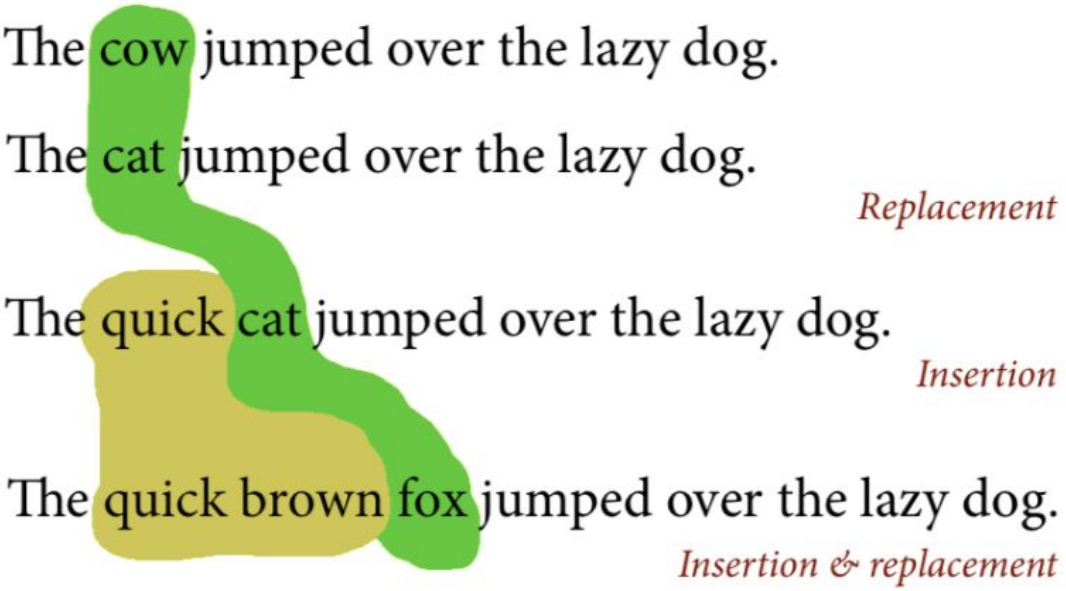
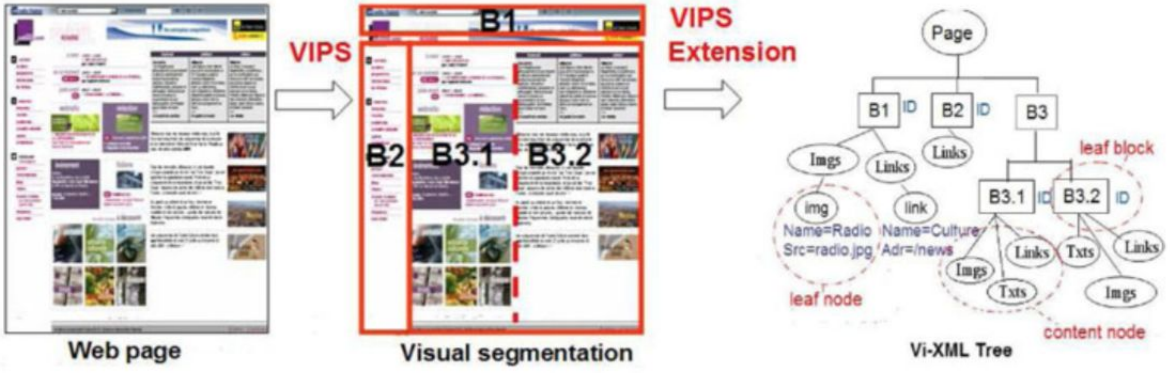


Figure 2: A sample text undergoing replacement and insertion operations. The highlighted areas track segments of the text that are affected by edits in consecutive versions.

Figure di [SQN10]

Un altro algoritmo che si occupa di identificare e mostrare le differenze di un documento, in particolare in questo caso di pagine web, è Vi-Diff [PBG10]. Quello proposto dagli autori è un metodo di rilevazione dei cambiamenti nelle versioni di pagine web. Individua delle differenze semantiche tra due versioni basandosi sulla loro rappresentazione visiva. Effettua quindi un analisi di ciò che è cambiato visivamente nella pagina web, focalizzandosi sulla struttura più che sul contenuto. Difatti il processo di analisi, che consiste in più passaggi, distingue fra quelle che sono delle modifiche a livello strutturale e quelle che sono le modifiche al contenuto. Dopo aver effettuato una segmentazione della pagina e dopo aver eseguito un'identificazione delle modifiche, produce un file contenente il delta (cioè la differenza tra le due versioni) della pagina web.
Questa modalità di analisi viene effettuata poiché secondo gli autori un semplice elenco di modifiche del contenuto di una pagina web non fornisce informazioni rilevanti su cosa è cambiato a livello di significato. Effettuano invece un'analisi ad un livello di astrazione più elevato rispetto a molti altri algoritmi di diffing e forniscono quindi informazioni utili e sfruttabili in diversi ambiti di applicazione.




```
<xml>
  <Page url="www.radiofrance.fr" version="V_01-10-09">
    <Block Ref="B1" ID="001A" Pos="H:10-W:20">
      <Links ID="012C" IDList="042C">
        <link ID="1L23" Name="Culture" Adr="/news">
      </Links>
      <Imgs ID="g15K" IDList="g25K">
        <img ID="25jL" Name="Radio" Src="/radio.jpg"/>
      </Imgs>
    </Block>
    <Block Ref="B2" ID="150K" Pos="H:53-W:10">
      <Links ID="1k2M" IDList="4k2M"> ... </Links>
    </Block>
    <Block Ref="B3" ...>
      <Block Ref="B3.1" ...> ... </Block>
      <Block Ref="B3.2" ...> ... </Block>
    </Block>
  </Page>
</xml>
```

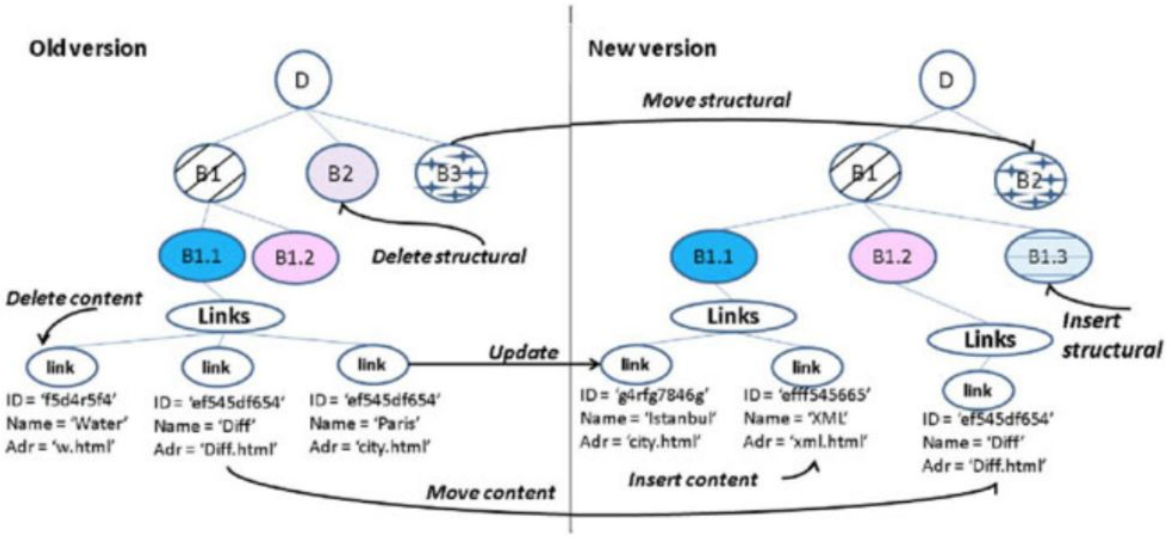


Fig. 2. Operations

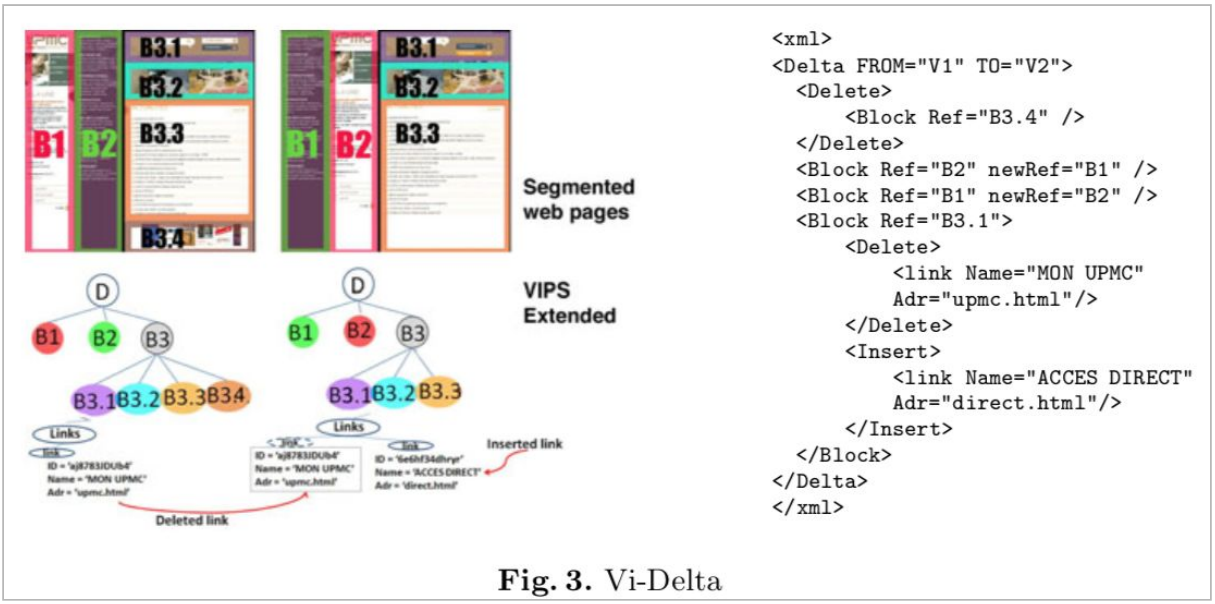


Fig. 3. Vi-Delta

Figure di [PBG10]

Un altro algoritmo che si occupa della visualizzazione di edits è DiffViz, creato da Frick, Wedenig e Pinzger [FWP18]. In quanto tool di visualizzazione di edits, DiffViz evidenzia le modifiche tra due versioni di un file attraverso una comparazione side-by-side, con a sinistra la vecchia versione ed a destra la nuova. Le modifiche vere e proprie sono rappresentate attraverso markup colorato, cioè visivamente parlando sono evidenziate con diversi colori. Vi sono tre diverse modalità di input per le modifiche: la prima è rappresentata da un inserimento manuale attraverso un editor integrato, la seconda permette di importare i file direttamente da GitHub, mentre la terza è formata dall'uso di un file JSON che fa riferimento a una o più coppie di diff GitHub. Una volta importati i file o le liste di modifiche, è possibile decidere quale algoritmo di diffing utilizzare tra i disponibili nella piattaforma e visualizzare i risultati nella modalità side-by-side. All'interno dell'articolo è riportato un esempio significativo ad illustrare quella che anche secondo me è una limitazione di molti tool di visualizzazione di diff, in particolare in questo caso il tool di GitHub: nel caso preso in esame dagli autori è presente una modifica categorizzabile come uno spostamento di alcune righe di codice in un'altra posizione all'interno del documento; DiffViz la riconosce giustamente come tale, mentre invece il tool di visualizzazione di GitHub non la riconosce come tale ma come una serie di cancellazioni e inserimenti. Da un punto di vista "meccanico", quest'ultima

rappresentazione non è errata, ma da un punto di vista semantico è una rappresentazione povera di significato e quindi da considerarsi sbagliata.
I limiti evidenti della piattaforma DiffViz sono la sua dipendenza in qualche modo a GitHub e la sua focalizzazione esclusivamente su file di codici sorgenti di software. Se la piattaforma fosse sviluppata in modo da poter analizzare qualsiasi tipo di documento avrebbe delle potenzialità molto più ampie ed una grande utilità.

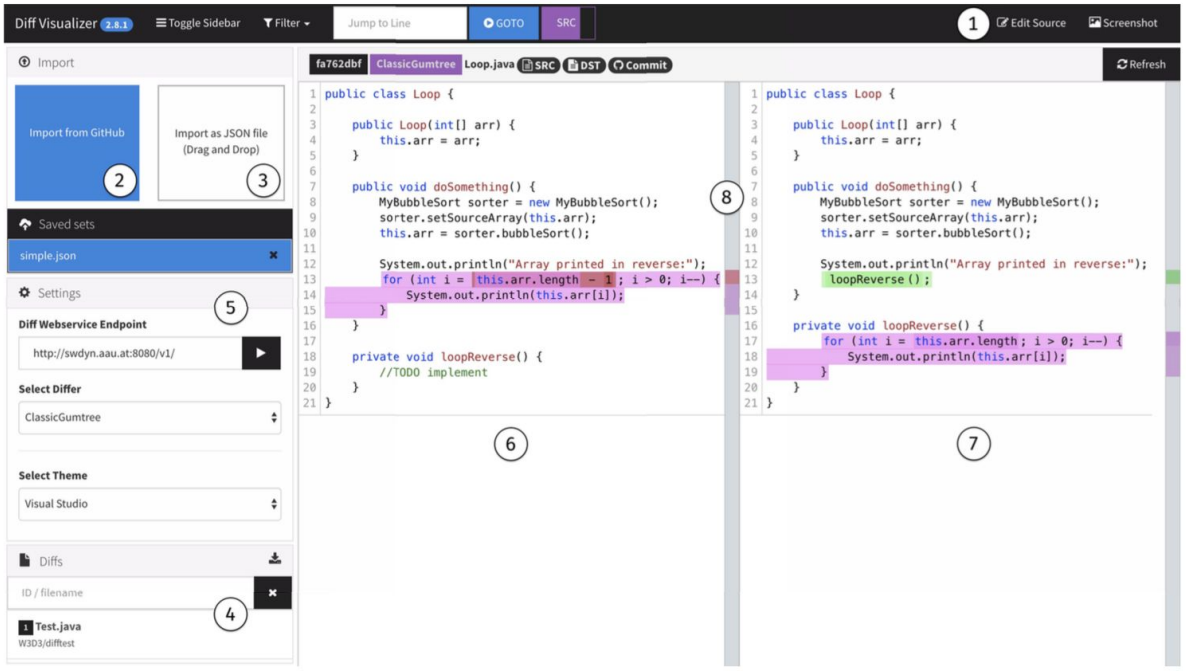


Fig. 1. Screenshot of DiffViz showing two versions of the Java class `Loop` where a bug was introduced in the newer version

Figura di [FWP18]

Un altro lavoro affine ad alcuni concetti chiave che compongono la mia tesi, nonostante non si occupi di visualizzazione di modifiche ma di comparazione di documenti e quindi di creazione di una lista di modifiche, è "diffi", un progetto di Barabucci [BA18]. Lo scopo di questo tool è descrivere le differenze tra il contenuto di due documenti, senza riguardo per il formato. L'autore si è prefissato di riuscire a creare un algoritmo che, prendendo in input due documenti anche di formato diverso, riuscisse a fornire come output una lista di modifiche che descriva a diversi livelli di astrazione le variazioni fra il contenuto dei due documenti.
L'ultimo progetto è PhiloEditor [IVD14], una piattaforma web che si propone di studiare, analizzare e mostrare le variazioni semantiche di testi, nel caso specifico del

romanzo “I Promessi Sposi” di Alessandro Manzoni. La creazione di questo progetto è nato dalla ricerca di un punto di incontro tra le necessità filologiche ed umanistiche di analisi dei testi e le tecnologie messe a disposizione dalla scienza del computer nel version management e nel change tracking dei sistemi di gestione dei documenti. Si è tentato, cioè, di digitalizzare il lavoro di analisi classico della filologia, semplificando il lavoro di critica delle varianti di documenti e testi.

Come già discusso in precedenza, gli autori illustrano all’interno dell’articolo di presentazione del progetto che esistono varie tipologie di algoritmi di diffing, le quali si differenziano, tra gli altri aspetti, anche sulla tipologia di delta che producono. Quelli tipicamente usati nella comparazione di file sorgenti di software si basano sulla ricerca di modifiche a livello di righe di codice, in quanto nei linguaggi di programmazione esse sono gli elementi fondamentali, semanticamente parlando, del processo di produzione del programma. Ma nell’ambito di studio relativo a PhiloEditor, gli autori hanno trovato essere più appropriato un algoritmo di tipo word-based, cioè un algoritmo basato sulla ricerca di variazioni al livello delle parole.

La piattaforma presenta le differenze fra due versioni dei Promessi Sposi, quella definita “Ventisettana” e quella definita “Quarantana”, cioè la versione del 1827 e quella del 1840, evidenziando attraverso l’uso di stili e colori diversi le varie tipologie di modifiche effettuate dal Manzoni. Oltre a mostrare le operazioni meccaniche di inserimento e cancellazione, categorizza in maniera semantica alcune di queste operazioni meccaniche, arricchendo di significato le varianti. Ad esempio, vengono evidenziate in maniera differente le varianti di toscanizzazione da quelle che sono invece degli abbassamenti linguistici.

La limitazione più evidente che ho trovato e alla quale ho provato a proporre una soluzione attraverso la mia tesi è la staticità della piattaforma. Le variazioni non vengono generate dinamicamente ed è possibile solo analizzare i documenti all’interno della piattaforma.

Figure 1. Showing differences between two variants in PhiloEditor.

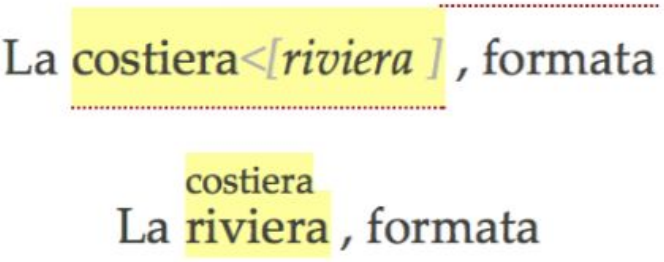


Figure 2. Zooming on a text modification. The differences can be shown in horizontal or vertical mode.

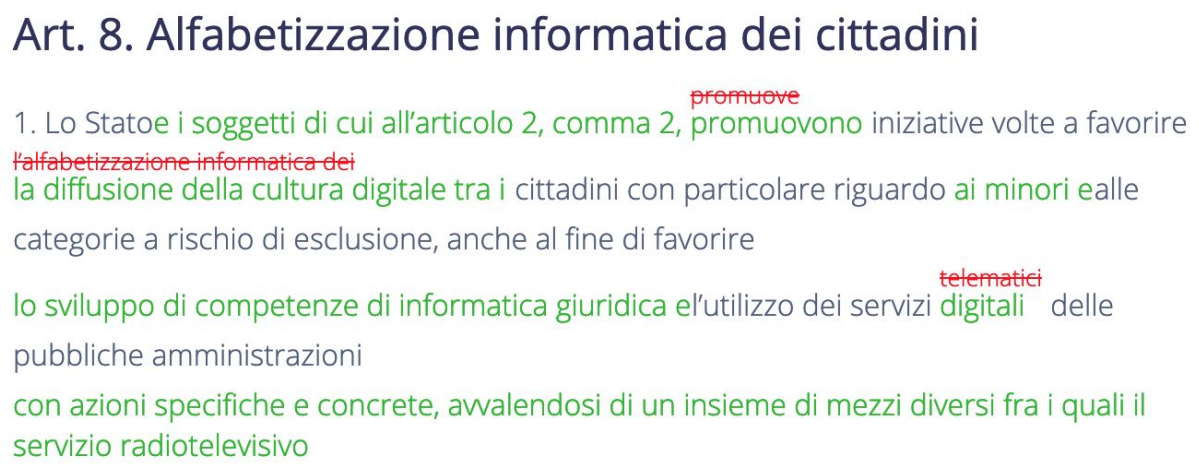
Figure di [IVD14]

3. Differ

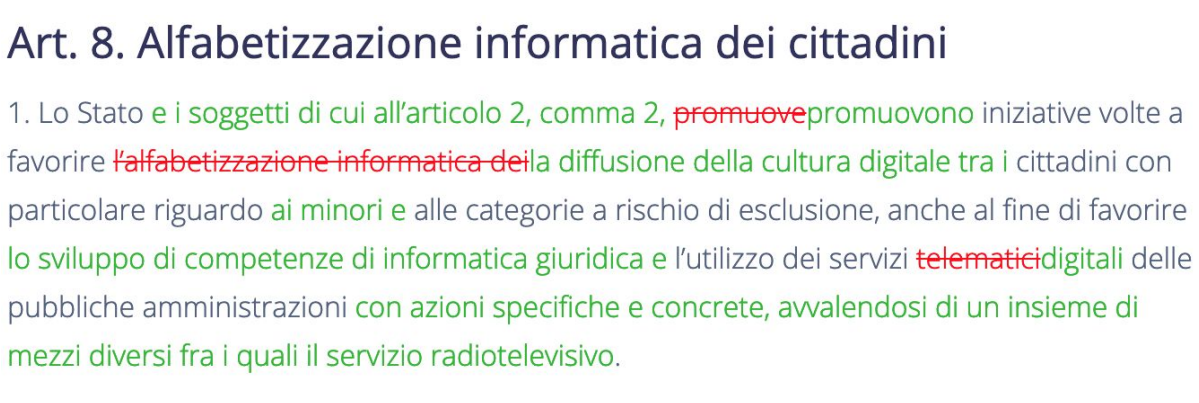
Alla luce dei limiti evidenziati dall'analisi del contesto scientifico e tecnologico dei servizi esistenti di change tracking e visualizzazione di modifiche a documenti, ho ideato e realizzato una libreria per i servizi web che ha lo scopo di riempire le lacune risultanti attraverso la categorizzazione e semantizzazione delle modifiche ed una loro visualizzazione efficace e qualificante.

3.1 Descrizione del sistema

Differ è una libreria Javascript per la visualizzazione di modifiche a documenti. È basata su una struttura dati per le modifiche a tre livelli, che permette il raggruppamento e la categorizzazione di esse. Nella prossima sezione spiegherò in dettaglio l'architettura sia della struttura dati che della libreria stessa.
Lo scopo di Differ è quello di offrire un servizio che permetta una visualizzazione strutturalmente e semanticamente consapevole delle modifiche ad un documento e anche un sistema di analisi e conservazione delle varie versioni di esso. Offre tre stili di visualizzazione, che ho chiamato Word style, Wiki style e Git Style: il primo compone, a partire da una prima versione di un documento e da un file con le modifiche, il documento finale con le modifiche effettuate, appositamente strutturate all'interno del testo, una lista di contatori delle tipologie di modifiche eseguite e una lista di regole grafiche da applicare, completamente o in parte, per ottenere un'interfaccia grafica che mostra i cambiamenti in due modalità: orizzontale, dove inserimenti e cancellazioni sono visualizzati linearmente, e verticale, dove nei casi in cui si abbiano delle operazioni di tipo replace, il contenuto eliminato viene mostrato nell'interlinea del testo, in corrispondenza del contenuto inserito al suo posto.



Word style con visualizzazione verticale delle modifiche

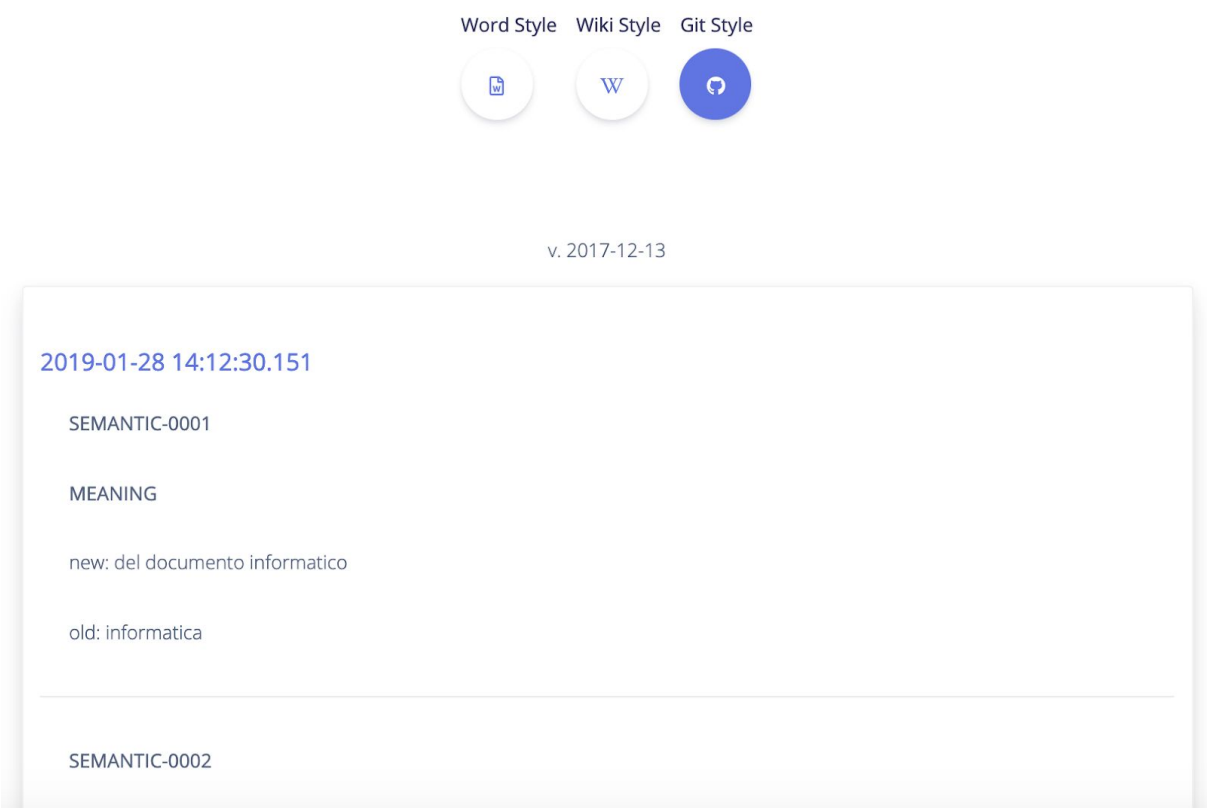


Word style con visualizzazione orizzontale delle modifiche

Il secondo stile, invece, mostra le modifiche in due colonne affiancate, una dedicata alle cancellazioni e l'altra agli inserimenti, in modo che nel caso di operazione singola, la casella corrispondente all'operazione complementare sia vuota, mentre nel caso di operazione multipla, come nel caso di un replace, le due operazioni meccaniche siano mostrate in corrispondenza una dell'altra.

Wiki style: contenuto dell'operazione compresso (sopra) ed esteso (sotto)

L'ultimo stile, infine, mostra le modifiche raggruppate temporalmente, in base al timestamp della edit.



Git style

Questi stili di visualizzazione ottenuti dai risultati della libreria permettono di facilitare la comprensione delle modifiche ad un documento, caratterizzando la struttura degli elementi che le definiscono, e permettono anche di mantenere una traccia delle varie versioni di un documento.

Per mostrare il funzionamento della libreria, ho creato anche una piattaforma web che effettua le chiamate alle varie funzioni di Differ ed elabora i risultati, mostrando all'utente un esempio di caso d'uso.

In particolare, ho utilizzato come documento il CAD, il Codice dell'Amministrazione Digitale, un atto normativo della Repubblica Italiana, precisamente il decreto legislativo 7 marzo 2005, n. 82. Questo testo costituisce un corpo organico di disposizioni che presiedono all'uso dell'informatica come strumento privilegiato nei rapporti tra la pubblica amministrazione italiana e i cittadini dello stato. E' un testo piuttosto lungo (circa 400 pagine) e molto strutturato. Essendo un codice, specie su una materia soggetta a continua evoluzione tecnologica, è un testo normativo

periodicamente aggiornato, di conseguenza un esempio calzante a dimostrazione di un possibile utilizzo di Differ.
Difatti vi sono molteplici versioni del CAD, spesso con rilevanti differenze tra loro, e una elaborazione di esse attraverso l'uso dei servizi messi a disposizione dalla mia libreria è utile ad analizzare lo sviluppo storico del testo normativo.
In particolare, ho analizzato due versioni di esso, quella del 13 Dicembre 2017 e quella del 18 Ottobre 2012, entrambe a partire dalla versione originale del 7 Marzo 2005.

3.2 Servizi della libreria

Vi sono tre servizi principali di Differ, i quali forniscono i dati necessari per effettuare le tre diverse visualizzazioni da me elaborate. Nel prossimo capitolo fornirò un'analisi dettagliata di come le edits vengono elaborate e dei diversi oggetti risultanti dalle chiamate ai servizi dedicati ai vari tipi di visualizzazione.
Il primo servizio, orientato al Word style, fornisce il documento elaborato con le modifiche apportate ad esso, una lista del numero di operazioni effettuate per tipologia e una serie di regole CSS da applicare per ottenere specialmente la visualizzazione verticale, in quanto quelle relative alla visualizzazione orizzontale offrono esclusivamente una colorazione e un'evidenziazione delle modifiche, le quali potrebbero essere differenti a discrezione dell'utente. Questa funzione permette quindi di visualizzare il documento nella sua interezza, arricchito di tutte le modifiche subite, e di avere un resoconto sulla quantità e sulla tipologia di cambiamenti.

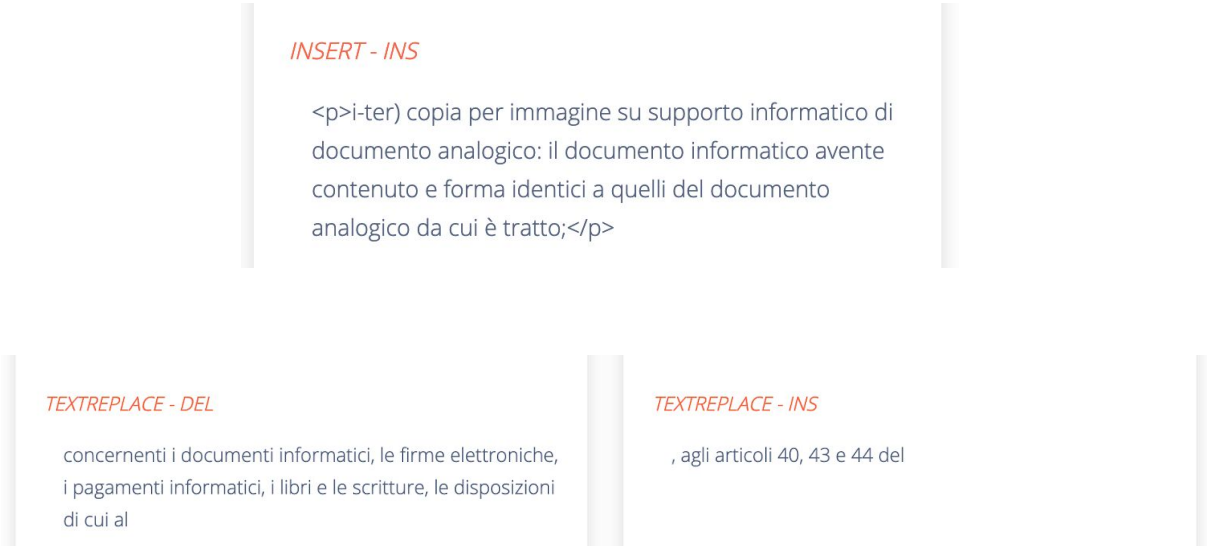
s) firma digitale: un particolare tipo di firma ~~elettronica~~ qualificata basata su un sistema di chiavi crittografiche, una pubblica e una privata, correlate tra loro, che consente al titolare tramite la chiave privata e ~~al destinatario~~a un soggetto terzo tramite la chiave pubblica, rispettivamente, di rendere manifesta e di verificare la provenienza e l'integrità di un documento informatico o di un insieme di documenti informatici;

Visualizzazione orizzontale

s) firma digitale: un particolare tipo di firma ~~elettronica~~qualificata basata su un sistema di chiavi crittografiche, una pubblica e una privata, correlate tra loro, che consente al titolare tramite la chiave privata e ~~al destinatario~~ a un soggetto terzo tramite la chiave pubblica, rispettivamente, di rendere manifesta e di verificare la provenienza e l'integrità di un documento informatico o di un insieme di documenti informatici;

Visualizzazione verticale

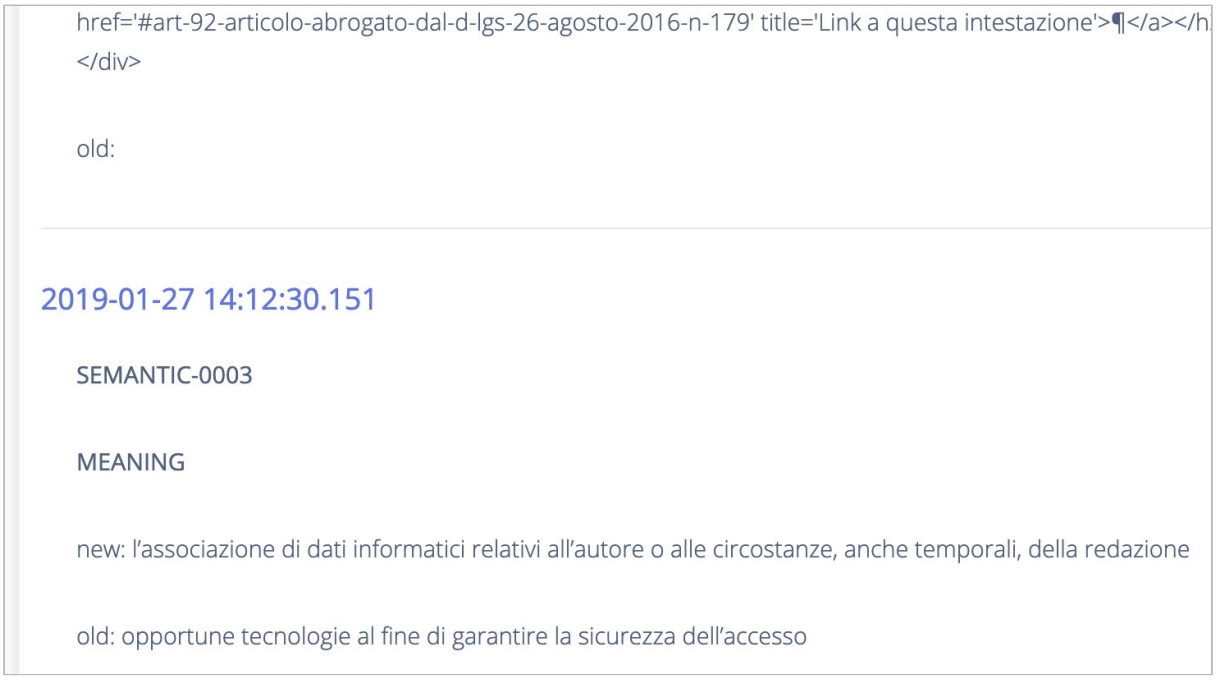
Il secondo servizio, orientato al Wiki style, offre una lista di operazioni strutturali, ognuna delle quali contiene le operazioni meccaniche di inserimento e cancellazione che la compongono e la tipologia di operazione strutturale (ad esempio, "INSERT" o "TEXTREPLACE").



Esempio Wiki style sulla piattaforma web

Il terzo e ultimo servizio, quello orientato al Git style, ritorna le edits contenute nella struttura dati raggruppate temporalmente in base al timestamp dell'operazione

semantica. Vengono raggruppate in base al timestamp e non, ad esempio, in base alla data poiché vi sono due casi d'uso principali nella creazione della struttura dati delle modifiche: che esse vengano generate a partire da due versioni del documento, e in questo caso il timestamp di ogni operazione sarà lo stesso perché vengono elaborate nello stesso momento, oppure che esse vengano generate ed aggiunte ad una struttura dati esistente, e che quindi siano differenziate anche nei millisecondi, in modo da poter tenere traccia del reale ordine cronologico delle modifiche e dello sviluppo temporale del documento.



Esempio Git style sulla piattaforma web

4. Architettura di Differ

Passando all'analisi dell'architettura di Differ, inizialmente illustrerò la struttura dati delle edits ed in seguito l'architettura della libreria Javascript. Infine, descriverò come ho elaborato i risultati della libreria per creare le visualizzazioni sulla piattaforma web che ho realizzato.

4.1 Struttura dati delle edits

Come già detto in precedenza, le modifiche ad una versione di un documento vengono descritte attraverso una struttura dati particolare, la quale mi è stata presentata nel momento in cui ho iniziato a sviluppare il progetto.
Si tratta di un oggetto JSON articolato su tre livelli: semantico, strutturale e meccanico. Ogni livello contiene informazioni diverse inerenti alle modifiche da effettuare su un documento.
A livello meccanico troviamo le modifiche testuali, con informazioni relative al tipo di operazione, alla posizione nel documento e al contenuto. I due tipi di operazione possibili a questo livello sono l'inserimento e la cancellazione, definite come INS e DEL. Queste operazioni agiscono esclusivamente a livello della stringa di testo, nonostante comprendano anche operazioni che agiscono sul markup del documento. Per quel che riguarda la posizione, viene calcolata sulla stringa dei caratteri del testo e tiene conto di ogni tipo di carattere presente, anche il whitespace. Si può quindi dire che sono operazioni atomiche, che prese singolarmente non hanno significati particolari se non quello di tracciare modifiche di caratteri.
A livello strutturale troviamo le prime informazioni sul tipo di modifica che viene effettuata. Le operazioni strutturali sono suddivise in operazioni sul testo e operazioni sulla struttura. Le prime, quelle sul testo, si risolvono all'interno di un unico nodo di testo, mentre le seconde si risolvono su strutture di markup complesse. Quindi, le operazioni di testo avvengono all'interno di una parola, oppure su una o più parole; e possono essere le seguenti:

- TEXTREPLACE

  Si tratta di una sostituzione di una stringa di più parole diverse o di un intero periodo con un'altra stringa o periodo, all'interno dello stesso nodo. È quindi composta da un inserimento e un'eliminazione nella stessa posizione ma con contenuto diverso.


```
{
 "id": "STRUCTURAL",
 "new": "del domicilio digitale",
 "old": "della posta elettronica certificata",
 "operation": "TEXTREPLACE",
 "items": [{
     "id": "EDIT",
     "operation": "DEL",
     "position": 15021,
     "content": "della posta elettronica certificata"
        }, {
     "id": "EDIT",
     "operation": "INS",
     "position": 15021,
     "content": "del domicilio digitale"
 }]}
```

- WORDCHANGE

  Si tratta di modifiche di una singola parola senza variazione della parola successiva (ad esempio, “giuoco” con “gioco”).


```
{

 "id": "STRUCTURAL",

  "new": "modalità",

  "old": "Modalità",

 "operation": "WORDCHANGE",

 "items": [{

     "id": "EDIT",

     "operation": "DEL",

     "position": 29837,

     "content": "M"

 }, {

     "id": "EDIT",

     "operation": "INS",

     "position": 29837,

     "content": "m"

     }]}
```

- WORDREPLACE

  Si tratta di modifiche dove un'intera parola viene sostituita con un'altra. In questo caso con parola si intende una unità atomica di significato, formata quindi da un sostantivo con il suo eventuale articolo ed un eventuale aggettivo determinativo, oppure da un verbo e il suo eventuale verbo ausiliare.


```
{
 "id": "STRUCTURAL",
 "new": "trasferimenti",
 "old": "pagamenti",
 "operation": "WORDREPLACE",
 "items": [{
     "id": "EDIT",
     "operation": "DEL",
     "position": 35678,
     "content": "pagamenti"
 }, {
     "id": "EDIT",
     "operation": "INS",
     "position": 35678,
     "content": "trasferimenti"
 }]}
```

- TEXTINSERT

  Si tratta di un inserimento di una stringa, che può essere composta da una o più parole, escluso il markup.


```
{

 "id": "STRUCTURAL",

 "new": " per la formazione,",

"old": "",

 "operation": "TEXTINSERT",

 "items": [{

     "id": "EDIT",

     "operation": "INS",

     "position": 36668,

     "content": " per la formazione,"

 }]}
```


- TEXTDELETE

  Complementare alla precedente: al posto di un inserimento si trova una eliminazione.


```
{

 "id": "STRUCTURAL",

 "new": "",

 "old": "temporale ",

 "operation": "TEXTDELETE",

 "items": [{

     "id": "EDIT",

     "operation": "DEL",

     "position": 36758,

     "content": "temporale "

 }]}
```

- PUNCTUATION

  Si tratta di modifiche della sola punteggiatura che non incidono in alcuna maniera nel testo vero e proprio. Sono incluse anche le modifiche complesse che prevedono, ad esempio, una modifica alla parola o carattere successiva causata dall'operazione di punteggiatura.


```
{
 "id": "STRUCTURAL",
 "operation": "PUNCTUATION",
 "items": [{
     "id": "EDIT",
     "operation": "INS",
     "position": 65207,
     "content": ","
 }]}
```


Le operazioni sulla struttura, invece, sono quelle che avvengono tra più nodi di markup, a volte comprendono solo caratteri di markup, altre volte anche il testo contenuto o intorno al markup.
Possono essere:

- REPLACE

  Si tratta della sostituzione di un nodo con un altro nodo. È quindi formata da inserimenti e rimozioni di nodi nella stessa posizione.


```
{

 "id": "STRUCTURAL",

  "new": "Agli effetti del presente decreto legislativo si intende per:<p>a) formato dei dati di tipo aperto[...]",

 "old": "Per formato dei dati di tipo aperto si intende[...]",

 "operation": "REPLACE",

 "items": [{

     "id": "EDIT",

     "operation": "DEL",

     "position": 121145,

     "content": "Per formato dei dati di tipo aperto si intende[...]"

 }, {

     "id": "EDIT",

     "operation": "INS",

     "position": 121145,

       "content": "Agli effetti del presente decreto legislativo si intende per:<p>a) formato dei dati di tipo aperto[...]"

 }]}
```

- MOVE

  Si tratta dello spostamento di uno o più nodi all'interno del documento. È quindi composta da inserimenti e rimozioni con lo stesso contenuto ma in posizioni diverse.


```
{
 "id": "STRUCTURAL",

 "operation": "MOVE",

 "items": [{

     "id": "EDIT",

     "operation": "DEL",

     "position": 121039,

     "content": "<p>Per formato dei dati di tipo aperto si intende[...]</p>"

 }, {

     "id": "EDIT",

     "operation": "INS",

     "position": 121039,

     "content": "<p>Per formato dei dati di tipo aperto si intende[...]</p>"

 }]}
```

- WRAP

  Si tratta di annidamento di uno o più nodi con un nuovo nodo. È quindi composta da più inserimenti esclusivamente di markup. Non cambia il contenuto ma il livello strutturale.

```
{
 "id": "STRUCTURAL",

 "new": "<b>basata</b>",
 "old": "basata",
 "operation": "WRAP",
 "items": [{
     "id": "EDIT",
     "operation": "INS",
     "position": 123000,
     "content": "<b>"
 }, {
     "id": "EDIT",
     "operation": "INS",
     "position": 123011,
     "content": "</b>"
 }]}
```

- UNWRAP

  È l'operazione inversa alla precedente: viene rimosso un nodo mantenendo il suo contenuto ed è quindi composta da più rimozioni esclusivamente di markup. Anche in questo caso non cambia il contenuto ma il livello strutturale.


```
{
 "id": "STRUCTURAL",

 "new": "basata",

 "old": "<b>basata</b>",

 "operation": "UNWRAP",

 "items": [{

     "id": "EDIT",

     "operation": "DEL",

     "position": 123000,

     "content": "<b>"

 }, {

     "id": "EDIT",

     "operation": "DEL",

     "position": 123011,

     "content": "</b>"

 }]}
```

- JOIN

  Si tratta dell'unione di due nodi di testo fratelli e dello stesso tipo. Tecnicamente, è composta da due cancellazioni, anche qui esclusivamente di markup.


```
{"id": "STRUCTURAL",

 "new": "<p>Alla luce [...] è inattendibile</p>",

 "old": "<p>Alla luce [...] </p><p>è inattendibile</p>",

 "operation": "JOIN",

 "items": [{

     "id": "EDIT",

     "operation": "DEL",

     "position": 11263,

     "content": "</p><p>"

 }]}
```


- SPLIT

  Complementare al JOIN, si tratta della separazione di un nodo in due nodi fratelli dello stesso tipo. È quindi composta da due inserimenti esclusivamente di markup.


```
{

 "id": "STRUCTURAL",

 "new": "<p>Alla luce [...] </p><p>è inattendibile</p>",

 "old": "<p>Alla luce [...] è inattendibile</p>",

 "operation": "SPLIT",

 "items": [{

     "id": "EDIT",

     "operation": "INS",

     "position": 11263,

     "content": "</p><p>"

 }]

}
```

- INSERT

  Si tratta dell'inserimento di uno o più nodi di testo ed eventualmente del testo prima e/o dopo ai nodi.


```
{
 "id": "STRUCTURAL",
   "new": "<p>6. Per quanto non previsto dal presente articolo si applicano gli articoli 21, 22 , 23 e 23-bis.</p>",
 "operation": "INSERT",
 "items": [{
     "id": "EDIT",
     "operation": "INS",
     "position": 44408,
        "content": "<p>6. Per quanto non previsto dal presente articolo si applicano gli articoli 21, 22 , 23 e 23-bis.</p>"
 }]}
```

- DELETE

  Operazione complementare alla precedente, con cancellazioni invece che inserimenti.


```
{
 "id": "STRUCTURAL",
     "old": "<p>d) sull’adempimento degli obblighi a suo carico previsti dall’articolo 32.</p>",
 "operation": "DELETE",
 "items": [{
     "id": "EDIT",
     "operation": "DEL",
     "position": 58661,
      "content": "<p>d) sull’adempimento degli obblighi a suo carico previsti dall’articolo 32.</p>"
 }]}
```

- NOOP

  Sono modifiche non rilevanti, che non hanno alcun effetto concreto. Possono essere all'interno di markup, come modifiche di ordine negli attributi di un elemento, oppure all'interno del testo, ad esempio l'inserimento o la rimozione di whitespace.

```
{
 "id": "STRUCTURAL",
 "new": "<section class='header' id='header-info'>",
 "old": "<section class='header'>",
 "operation": "NOOP",
 "items": [{
     "id": "EDIT",
     "operation": "INS",
     "position": 20154,
     "content": "id='header-info'"
 }]}
```

L'ultimo livello è quello semantico, dove le operazioni vengono descritte ad un livello più astratto ma anche in modo impreciso e soggettivo, quindi opinabile. Difatti sono semantiche quelle operazioni o sequenze di operazioni a cui si riesce a dare un significato immediatamente comprensibile ad un essere umano. La distinzione, essendo intuitiva, è quindi soggetta ad interpretazione e può essere discutibile. Inoltre non è facile determinare un elenco definitivo e preciso di categorie utilizzabili. A questo si aggiunge che alcune operazioni strutturali contengono già sufficienti informazioni che ne determinano un significato. Quindi in alcuni casi la caratterizzazione semantica viene usata per raggruppare operazioni strutturali in sovrastrutture; in altri casi viene usata solo come un'ulteriore modalità di descrizione di esse.

Per quel che riguarda questo livello di descrizione, le categorie individuate sono:

- FIX
  Si tratta di correzioni, ovvero operazioni strutturali che correggono un errore grammaticale, strutturale o concettuale.

- STYLE
  Si tratta di modifiche stilistiche, ovvero operazioni strutturali che cambiano l'aspetto e/o il contenuto senza modificarne il significato.

- MEANING
  In teoria, questa categoria è dedicata alle operazioni che cambiano il significato del testo. Nella pratica, è una categoria residuale: cioè ogni modifica alla quale non possiamo attribuire un significato semantico viene interpretata come un cambio di significato.

A queste si aggiungono alcune categorie di raggruppamenti semantici di operazioni strutturali:

- EDIT WAR
  Le cosiddette "guerre di modifiche", cioè modifiche ripetute, localizzate e il cui contenuto oscilla tra due o più stati diversi a cui si ritorna con una certa regolarità.

- EDIT WAKE
  Si tratta di una scia di modifiche, cioè una serie di piccole modifiche per ristrutturare e riorganizzare la frase o il testo in seguito ai problemi, anche banalmente grammaticali, causati da una prima modifica tipicamente "vera".

- EDIT CHAIN

  Si tratta di una catena di modifiche, cioè una serie di modifiche uguali o molto simili, spesso attivate in maniera automatica, come un cerca e sostituisci globale (ad esempio, nella stesura di un romanzo la decisione di cambiare il nome di un personaggio a metà dell'opera).

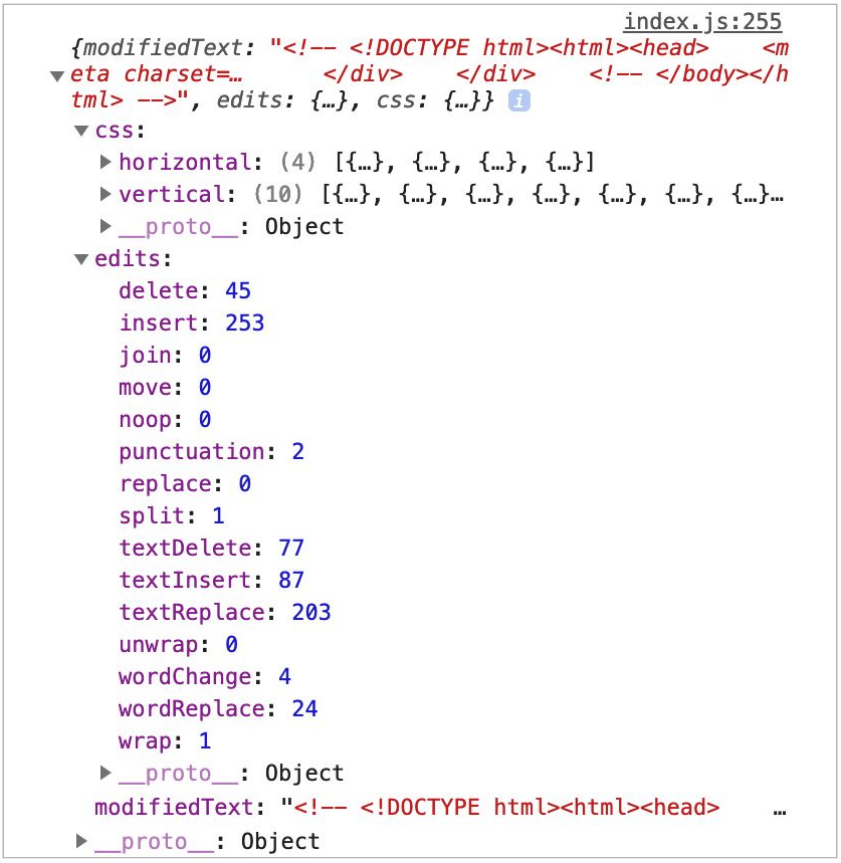
Esempi completi di edit

Per utilizzare la libreria Differ, l'utente deve avere a disposizione una prima versione del documento e uno o più file JSON contenenti la lista delle edits. Questi ultimi vengono generati da un'altra libreria, del quale non sono il creatore, la quale, una volta ricevuti in input due differenti versioni di un testo o documento, ne esegue un'analisi delle variazioni e crea il file JSON appositamente strutturato, dove le modifiche hanno una caratterizzazione semantica e strutturale.

4.2 Architettura della libreria

La libreria è composta da tre file Javascript: word.js, wiki.js e git.js, ognuno dedicato al rispettivo stile di visualizzazione.

In word.js vi sono un alto numero di funzioni utilizzate internamente, ma solo una esposta all'utente, wordStyle(), la quale riceve in input il documento originale sotto forma di stringa e il JSON con le modifiche da effettuare, e restituisce in output il documento finale formato, la lista di contatori delle tipologie di edits e le regole CSS.



Esempio di output di wordStyle()

Il meccanismo di questa funzione è il seguente: viene effettuata un'iterazione delle edit a rovescio, cioè a partire dall'ultima (per evitare una compromissione delle posizioni delle operazioni meccaniche), viene effettuato un controllo sul tipo di operazione ed in base a questo viene richiamato un comportamento differente. Per ora la funzione opera a livello strutturale, ma in una futura estensione dovrà operare a livello semantico. Per ogni operazione strutturale, viene effettuata un'altra

iterazione sulle operazioni meccaniche al suo interno, vengono salvate le informazioni relative alla posizione e al contenuto delle edits in Array, i quali vengono dati in input alla funzione dedicata ad eseguire il tipo di operazione strutturale incontrata. L'algoritmo è così ideato poiché non è possibile assumere a priori il numero di operazioni meccaniche all'interno di una strutturale, a parte il caso di TEXTINSERT, TEXTDELETE, INSERT e DELETE, che possono essere considerate delle operazioni residuali, dove per definizione si ha una sola operazione meccanica. Nei casi particolari sopra citati, viene comunque eseguita un'iterazione sulle operazioni meccaniche, per evitare errori nell'esecuzione, e per ogni edit viene richiamata la funzione apposita.
Le funzioni dedicate all'attuazione delle operazioni sono le seguenti:

- differDel(start, content, operation, author) - riceve in input la posizione, il contenuto, il tipo di operazione e l'autore ed è dedicata alle cancellazioni di tipo testuale.

- differDelStruct(start, content, operation, author) - riceve in input le stesse informazioni della precedente ma è dedicata alle cancellazioni di tipo strutturale.

- differDelUnwrap(start1, tag1, start2, tag2, author) - è dedicata esclusivamente all'operazione UNWRAP, e riceve in input le due posizioni dei due tag da rimuovere e gli stessi due tag, oltre all'autore.

- differDelWithoutWrap(pos, content) - è dedicata alle situazioni in cui non è possibile "avvolgere" le operazioni all'interno di markup. In questo caso vengono semplicemente eseguite, eliminando il contenuto dal testo; riceve in input la posizione e il contenuto della edit.

- differIns(pos, content, operation, author) - riceve in input posizione, contenuto, tipo di operazione ed autore, ed è dedicata agli inserimenti di tipo

testuale.

- differInsStruct(pos, content, operation, author) - riceve in input le stesse informazioni della precedente ma è dedicata agli inserimenti di tipo strutturale.

- differInsWrap(pos1, content1, pos2, content2, author) - è dedicata esclusivamente all'operazione WRAP, e riceve in input le due posizioni dei tag da inserire e il contenuto dei due tag, oltre all'autore.

- differInsSplit(pos, content, author) - è dedicata esclusivamente all'operazione SPLIT, e riceve in input la posizione, il contenuto e l'autore dell'edit.

- differInsWithoutWrap(pos, content, author) - come differDelWithoutWrap(), è dedicata a quei casi di inserimento in cui non è possibile "avvolgere" il contenuto all'interno di markup; prende in input la posizione, il contenuto e l'autore della edit.

- differReplace(pos, delContent, insContent, operation, author) - riceve in input array di posizioni, contenuti delle cancellazioni, contenuti degli inserimenti, e operazione ed autore delle edits; viene chiamata nei casi di PUNCTUATION, WORDCHANGE, WORDREPLACE e TEXTREPLACE.

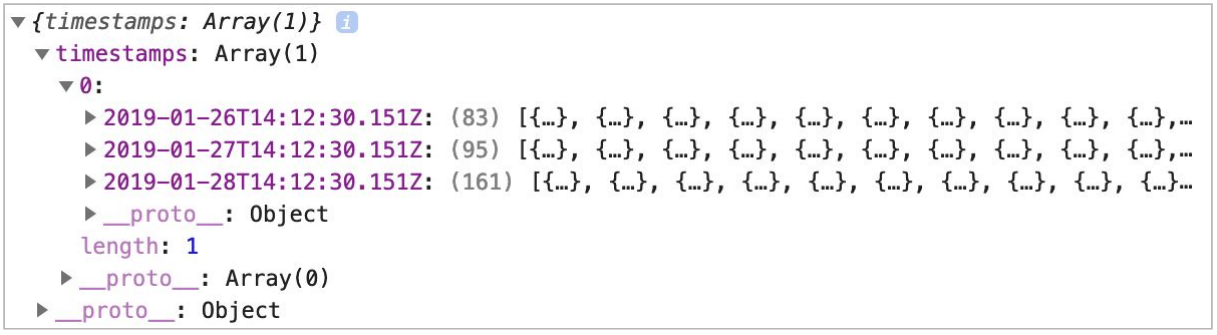
- differReplaceStruct(pos, delContent, insContent, operation, author) - simile alla precedente, ma gestisce solo REPLACE.

Il meccanismo base di ogni funzione è creare un elemento "contenitore" padre, uno span o un div a seconda del tipo di operazione o nodo da modificare, con un attributo custom "data-differ-diff", e un elemento figlio, anche in questo caso span o div a seconda del tipo di operazione, anch'esso con un attributo custom: "data-differ-del" nel caso di una rimozione e "data-differ-ins" nel caso di un inserimento. Inoltre, l'attributo custom dell'operazione meccanica ha come valore il tipo di operazione

eseguita. L'elemento padre è dotato anche di un tooltip informativo relativo all'autore. Dopo aver creato questa struttura, essa viene iniettata nel testo sfruttando le funzioni built-in di Javascript substring e slice, nella posizione appropriata.
Menzione speciale per alcune funzioni che invece hanno un meccanismo particolare. Le due funzioni dedicate al WRAP e UNWRAP agiscono in questo modo: differDelUnwrap() effettua un controllo sul tag da eliminare per decidere se usare span o div, poi effettua una cancellazione del contenuto con i tag ed un inserimento dello stesso contenuto senza i tag; differInsWrap() effettua anch'essa un controllo sul tag e poi effettua un inserimento del contenuto all'interno dei tag, e non solo dei due tag, che sono il reale contenuto delle operazioni meccaniche.
Le funzioni differDelWithoutWrap() e differInsWithoutWrap() eseguono rimozioni ed inserimenti senza l'uso della struttura contenitrice, in quanto vengono chiamate in situazioni nelle quali è necessario non modificare la struttura del testo, ad esempio l'aggiunta di un id ad un elemento oppure la modifica del valore di un attributo href in un anchor element.
La funzione differInsSplit() agisce inserendo l'attributo custom "data-differ-ins" con valore "structure SPLIT" nel secondo tag facente parte del contenuto dell'operazione meccanica, cioè quello di apertura.
Per quel che riguarda le regole CSS, in particolare quelle relative alla visualizzazione verticale, l'oggetto che le definisce contiene anche un campo "applyToParent" booleano, che indica se la corrispondente regola deve essere applicata all'elemento indicato oppure all'elemento padre.
In wiki.js, invece, vi è una sola funzione, wikiStyle(), la quale riceve in input il JSON delle edits. Su questa lista viene eseguita un'iterazione e, per ogni operazione strutturale, viene creato un oggetto che contiene: un indice incrementale, il contenuto dei campi old e new, l'operazione strutturale, e due array, uno per le rimozioni e uno per gli inserimenti, che immagazzinano il contenuto delle operazioni meccaniche, a seconda della loro tipologia. Una volta terminata l'iterazione, l'oggetto così formato viene fornito come output.

Esempio di output di wikiStyle()

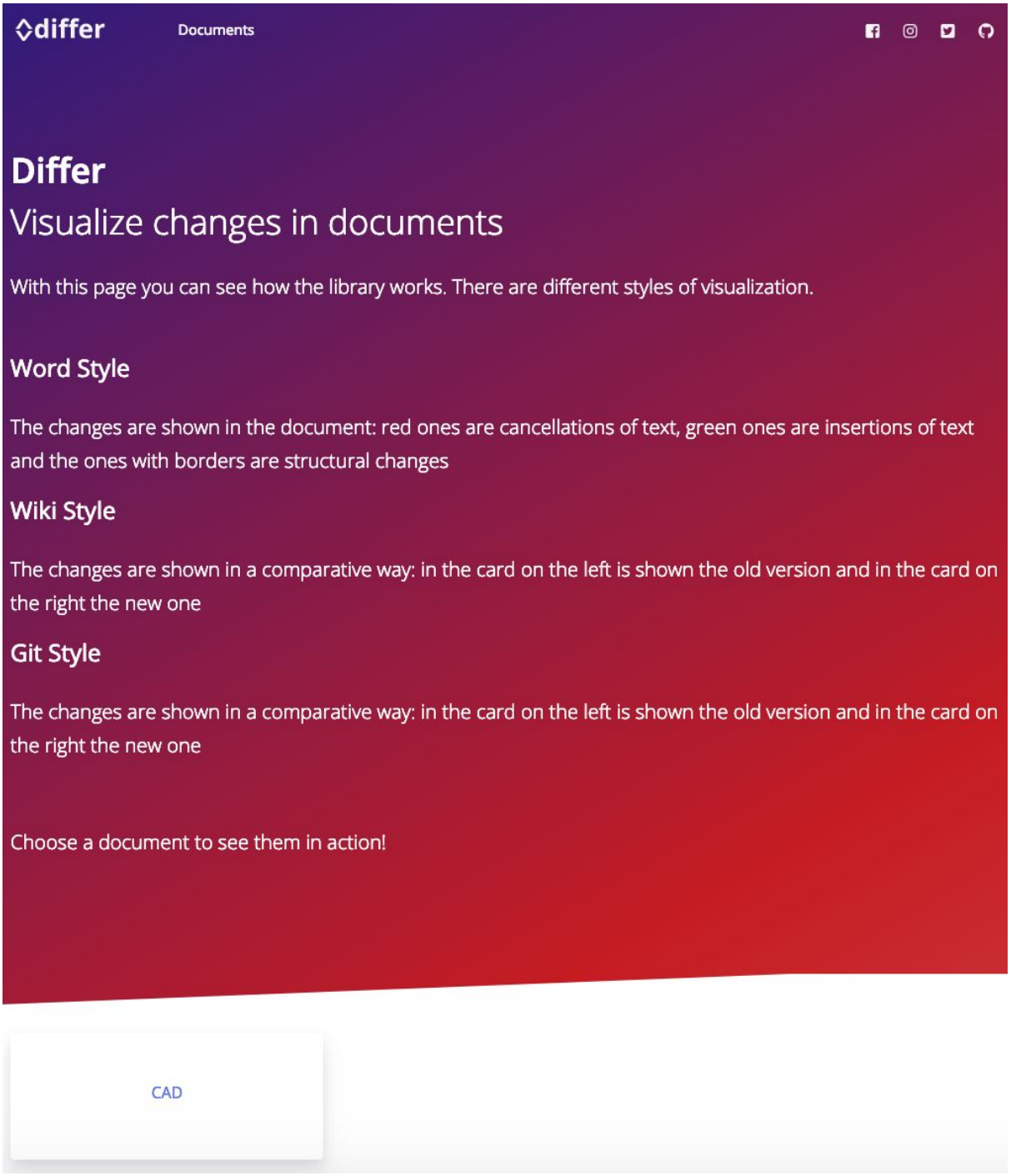
Infine, anche in git.js vi è una sola funzione, gitStyle(), e anch'essa riceve in input il JSON delle edits. Nuovamente viene eseguita un'iterazione, in questo caso al livello semantico, e le edits vengono raggruppate in base al timestamp. Si forma in questo modo un oggetto di array, dove vi è un indice per ogni timestamp, e ad ogni indice è presente un array di oggetti, i quali contengono le edits corrispondenti a quel timestamp.



Esempi di output di gitStyle()

4.3 Descrizione della piattaforma web

La piattaforma web è un ambiente che ho creato per mostrare possibili utilizzi della libreria, simulando un caso d'uso con due versioni del Codice dell'Amministrazione Digitale. Il sito è una Single Page Application, elaborato e modificato dal codice nel file index.js, diviso in tre sezioni principali: una navbar, un header informativo e una sezione centrale dedicata al contenuto elaborato.



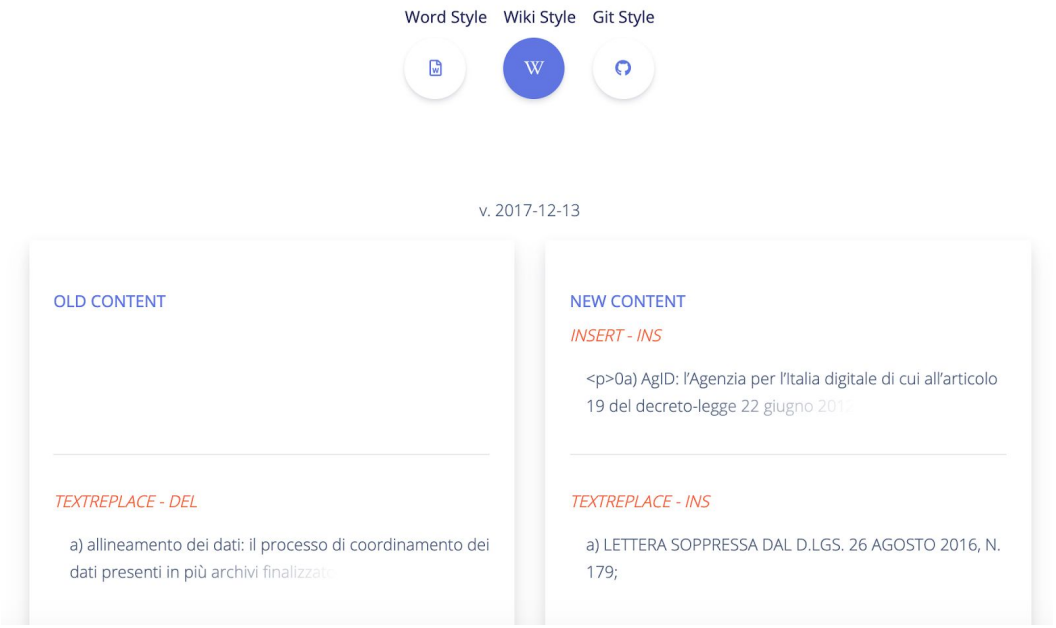
Pagina iniziale della piattaforma web di visualizzazione

La barra di navigazione ha un menù a cascata dal quale è possibile cambiare la versione del documento e alcuni link a social networks. L'header informativo illustra i tre stili possibili e si modifica a runtime in base allo stile visualizzato al momento.
La sezione centrale, invece, inizialmente mostra una card cliccabile per selezionare il documento, ma al momento l'unico disponibile è il CAD. Una volta scelto il

documento, viene caricato il Word style e i pulsanti che permettono di cambiare stile. L'interfaccia grafica dedicata a questo stile è composta da una sezione riassuntiva del tipo di edit della versione del documento e dal documento elaborato con le modifiche effettuate e visualizzate nel testo.

Word style

L’interfaccia dello stile Wiki è composta da due colonne, una dedicata ai contenuti eliminati e una dedicata ai contenuti inseriti, in una visualizzazione dove le operazioni meccaniche che compongono un’operazione strutturale sono allineate orizzontalmente e collegate. Ogni casella contiene informazioni sul tipo di operazione e sul contenuto di essa. Nei casi in cui il contenuto delle operazioni meccaniche sia lungo, viene inizialmente mostrato solo in parte, con la possibilità di visualizzarlo nella sua interezza cliccando sulla casella. Questa operazione aumenterà l’altezza anche della casella corrispondente all’operazione meccanica complementare.



Wiki style

In ultimo, l’interfaccia dello stile Git è composta da una sola colonna dove le operazioni vengono mostrate raggruppate in base al timestamp. Vi è quindi una sezione per ognuno di essi, e all’interno una casella per ogni operazione semantica, dove vengono mostrati l’id dell’operazione, il tipo ed i campi new e old, che indicano rispettivamente il contenuto inserito ed eliminato.

v. 2012-10-18

2019-01-28 14:12:30.151

SEMANTIC-0001

MEANING

new: del documento informatico

old: informatica

SEMANTIC-0002

MEANING

new: del documento informatico

old: dell'insieme di dati attribuiti in modo esclusivo ed univoco ad un soggetto, che ne distinguono l'identità nei sistemi informativi, effettuata

Git style

5. Conclusione e sviluppi futuri

Sebbene questa libreria sia stata creata per parte di un progetto per editing di documenti ingegneristici molto strutturati, è già utilizzabile da subito come libreria autosufficiente per la visualizzazione di edit nella struttura dati descritta ampiamente in precedenza.

Complessivamente la libreria è funzionale, nonostante necessiti di modifiche per l'integrazione con altre parti del progetto suddetto (prima tra tutte, il motore che genera i livelli strutturale e semantico delle edit meccaniche).

Ritengo che la struttura ideata per l'inserimento e di conseguenza la visualizzazione delle modifiche nel documento offra all'utente svariate possibilità di personalizzazione sfruttando le informazioni elaborate.

Vi sono però alcune modifiche necessarie al perfezionamento della libreria: in primis, è necessario gestire all'interno della struttura di visualizzazione anche le operazioni semantiche, quindi sia la loro elaborazione che la loro visualizzazione, in quanto per ora viene eseguita una rappresentazione efficace solo fino al livello strutturale. Poi saranno necessarie le modifiche relative all'integrazione con le altri parti del progetto, per andare a comporre il sistema completo.

Inoltre vi sono alcune possibili estensioni che non sono riuscito ad implementare, per ragioni di tempo, ma che sono interessanti e per le quali ho teorizzato una possibile implementazione. Un'interessante feature per la piattaforma web di visualizzazione potrebbe essere quella di creare un tour guidato all'interno dell'evoluzione cronologica del documento: mediante l'uso di bottoni si potrebbe guidare l'utente, spostandosi sulla parte del documento in base all'ordine temporale delle modifiche. In questo modo si fornirebbe un'ulteriore modalità di comprensione del processo di sviluppo del documento.

Un'altra possibile estensione del progetto potrebbe essere quella di trasformare la piattaforma web in un editor di testo e implementare una funzionalità di Operational Transformation (cioè di raccolta delle modifiche al momento stesso della loro esecuzione e di mantenimento dell'integrità del testo nelle situazioni di editing

contemporaneo di più persone). Al momento della rilevazione delle modifiche, esse verrebbero aggiunte al file JSON di edits, o ne verrebbe creato uno nuovo nel caso in cui si stesse lavorando su una nuova versione, permettendo l'immediata visualizzazione strutturata della modifica appena effettuata.

6. Riferimenti bibliografici

[AI19] Apple Inc., *Pages per Mac: Rilevare le modifiche in un documento di Pages*, Apple, 04/01/2019, https://support.apple.com/kb/PH23729?locale=it_IT.

[ASF18] Apache Software Foundation, *Tracking changes to a document*, Apache, ultima modifica il: 02/07/2018, https://wiki.openoffice.org/wiki/Documentation/OOoAuthors_User_Manual/Writer_Guide/Tracking_changes_to_a_document.

[BA18] Barabucci G. *diffi: diff improved; a preview*. In DocEng '18: ACM Symposium on Document Engineering 2018, August 28–31, 2018, Halifax, NS, Canada. ACM, New York, NY, USA. https://doi.org/10.1145/3209280.3229084

[CMT14] Coakley S.M., Mischka J. Thao C. *Version-Aware Word Documents*. DChanges '14, September 16, 2014, Fort Collins, CO, USA. http://dx.doi.org/10.1145/2723147.2723152

[ER09] Ektrand MD, Riedl JT. *rv you're dumb: Identifying Discarded Work in Wiki Article History*. WikiSym '09, October 25-27, 2009, Orlando, Florida, U.S.A. https://doi.org/10.1145/1641309.1641317

[FBA10] Fong PKF, Biuk-Aghai RP. *What Did They Do? Deriving High-Level Edit Histories in Wikis*. WikiSym '10, July 7-9, 2010, Gdańsk, Poland https://doi.org/10.1145/1832772.1832775

[FWP18] Frick V., Wedenig C., Pinzger M. *DiffViz: A Diff Algorithm Independent Visualization Tool for Edit Scripts*. 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 15/08/2018. http://dx.doi.org/10.1109/ICSME.2018.00081

[GO19] Google LLC. *Google Docs*, Google, visitato il: 11/02/2019, https://www.google.it/intl/it/docs/about/

[IVD14] Italia P., Vitali F., Di Iorio A. *Variants and Versioning between Textual Bibliography and Computer Science*. AIUCD 2014. 2014. https://doi.org/10.1145/2802612.2802614

[MC19a] Microsoft Corp., *Track changes in Word*, Microsoft, ultima modifica il: 08/02/2019, https://support.office.com/en-us/article/track-changes-in-word-197ba630-0f5f-4a8e-9a77-3712475e806a.

[MC19b] Microsoft Corp., *GitHub code review*, GitHub, visitato il: 11/02/2019, https://github.com/features/code-review/ (https://help.github.com/articles/about-comparing-branches-in-pull-requests/)

[NCK92] Neuwirth C.M., Chandhok R., Kaufer D.S., Erion P., Morris J., Miller D. *Flexible Diff-ing In A Collaborative Writing System*. CSCW 92 Proceedings, 01/11/1992, Toronto, Ontario, Canada. https://doi.org/10.1145/143457.143473

[PBG10] Pehlivan Z., Ben-Saad M., Gançarski S. *Vi-DIFF: Understanding Web Pages Changes*. DEXA 2010: Database and Expert Systems Applications pp 1-15. 2010. https://link.springer.com/content/pdf/10.1007%2F978-3-642-15364-8_1.pdf

[SA07] Sabel M. *Structuring Wiki Revision History*. WikiSym'07, October 21–23, 2007, Montréal, Québec, Canada. https://doi.org/10.1145/1296951.1296965

[SQN10] Shannon R., Quigley A., Nixon P. *Deep Diffs: Visually Exploring the History of a Document*. AVI '10, 25-29/05/2010, Rome, Italy. https://doi.org/10.1145/1842993.1843063

[VWD04] Viegas FB, Wattenberg M, Dave K,. *Studying Cooperation and Conflict between Authors with history flow Visualizations*. CHI 2004, April 24–29, 2004, Vienna, Austria. https://doi.org/10.1145/985692.985765

[WF19] Wikimedia Foundation, *Aiuto:Cronologia*, Wikipedia, ultima modifica il: 20/12/2018, https://it.wikipedia.org/wiki/Aiuto:Cronologia