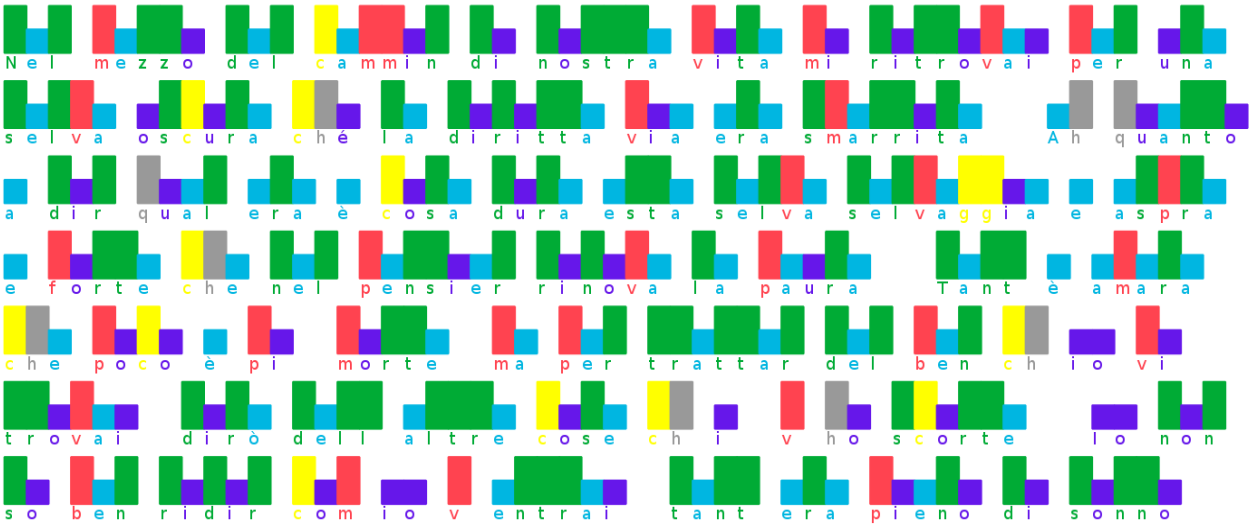
Stefano Penge

Lingua, programmi e creatività

Coding con le materie umanistiche



1. La teoria

1.1 Che cos'è il "coding"?

Per chi avesse bisogno di una rinfrescata veloce, "coding" è un termine inglese il cui significato è stato per anni “l’attività di scrivere codice sorgente”, ed è usato quasi come sinonimo di “programmare”. Quasi, perché programmare può significare anche *analizzare, progettare, verificare, integrare* un codice sorgente; mentre "coding" fa riferimento solo alla *scrittura* del codice, cioè alla trasformazione di un algoritmo in un testo in un determinato linguaggio di programmazione.

Curiosamente con “coding”, in questo momento storico, ci si riferisce invece alle attività di introduzione dei bambini alla programmazione, attraverso ambienti di programmazione *visuale*: cioè in cui paradossalmente non serve (anche se è possibile) *scrivere* il codice, ma è sufficiente *posizionare* oggetti simbolici che stanno al posto di operatori, variabili, condizioni.

Il MIUR, in collaborazione con il CINI, ha spiegato come e perché va introdotto il coding nella scuola in un sito dedicato al progetto "Programma il futoro".[1] L’obiettivo dichiarato è “fornire alle scuole una serie di strumenti semplici, divertenti e facilmente accessibili per formare gli studenti ai concetti di base dell’informatica”. L’iniziativa ripropone corsi e ambienti di lavoro creati e gestiti da un'associazione noprofit statunitense, Code.org, che ha come partner Microsoft, Google e tanti altri. Code.org ha in realtà obiettivi più ampi: punta all’integrazione razziale e a diminuire il gap di genere, a modificare i curricula delle scuole elementari e medie negli Stati americani, a soddisfare la richiesta di più informatica da parte dei genitori.

Con "Programma il Futuro" ritorna anche in Italia l’idea che si possa non solo *usare* la tecnologia digitale a scuola, ma anche *produrla*. Esaurita la spinta dei progetti ministeriali degli anni ottanta, in cui l’informatica come disciplina veniva inserita in vari modi nel curriculum, nei progetti degli anni seguenti il digitale era diventato ambiente e non contenuto; di conseguenza, l’idea di far *scrivere* programmi agli studenti era stata abbandonata. Oggi, sulla spinta di movimenti internazionali presenti in USA, Gran Bretagna, Francia, e sostenuti esplicitamente dai rispettivi Governi, si torna in qualche modo indietro: è ormai accettata l’idea che per cavarsela in un mondo sempre più digitale occorra sapere non solo come funziona l’app che usiamo per chattare, ma anche sapere come si crea. Per ragioni economiche, etiche, politiche: “Programma, o sarai programmato”. Chi non avrà queste competenze sarà un cittadino di seconda classe, costretto a subire le scelte degli altri. Il coding, insomma, è strettamente connesso all'acquisizione piena di quei diritti di cittadinanza digitale di cui si sente sempre di più la necessità.

Ora, al di là della condivisibilità dell’idea in sé, quello che colpisce in questo cambio di rotta è che il ruolo stesso della scuola viene a cambiare. La scuola, viene detto, si deve occupare subito di

1 http://programmailfuturo.it/come/ora-del-codice

questa urgenza futura; deve investire risorse non per educare nelle materie tradizionali tramite l'uso di tecnologie digitali, ma per proteggere i bambini da un futuro totalmente digitalizzato di cui non avranno la possibilità di essere attori. Questo è ben diverso da quanto ipotizzato nei vari Multilab, PNTD, Scuola 2.0: lì si parlava di didattica multimediale o di alfabetizzazione digitale – cioè in qualche modo si partiva da bisogni del presente - mentre ora si parte direttamente dal futuro. E' un passo importante, a cui potrebbe corrispondere un analogo interesse per il ruolo che avranno nel futuro le competenze linguistiche, matematiche, storiche che vengono acquisite oggi. Serviranno? Saranno fondamentali? Vanno aggiornate sulla base di come sarà la società tra dieci, venti anni? Domande importanti che meriterebbero una discussione più ampia di quella che potremmo imbastire qui.

Ci sono tante altre motivazioni che vengono utilizzate per promuovere il "coding": dall'apprendimento del pensiero computazionale[2] al divertimento che entra finalmente in classe, dal mutamento nel ruolo del docente fino al superamento del *gender gap*. E sono tante le discipline che vengono chiamate in causa: dalla matematica alla fisica alle scienze naturali.

Ma quella che brilla per la sua assenza è la lingua.

1.2 Coding e cultura umanistica

Questo testo presenta invece una serie di attività didattiche di "coding" proprio intorno ad argomenti di lingua. E' probabilmente l'unico, al momento, nel suo genere, non solo in Italia. L'idea è nata dalla rilettura di un testo rivoluzionario del 1989[3] in cui Giorgio Casadei, ordinario di Informatica a Bologna, presentava un percorso di apprendimento del Prolog dedicato a insegnanti di materie umanistiche. Se si poteva pensare di usare un linguaggio di programmazione (e uno davvero particolare come il Prolog) in una classe di Italiano trent'anni fa, perché non oggi che la programmazione è tornata in auge in modo così potente?

In realtà quel testo – fortemente innovativo proprio nella misura in cui attraversava un confine ritenuto invalicabile – era un vero manuale di Prolog e di programmazione logica, che intendeva accompagnare docenti e studenti delle superiori dai primi concetti alle tecniche di programmazione più avanzate. Gli esempi erano tratti dal mondo delle discipline umanistiche, ma il filo conduttore restava quello informatico, dell'apprendimento dello strumento. Le attività presentate – a mio avviso – oggi non sarebbero particolarmente interessanti di per se stesse (ad esempio, la costruzione di un database geografico) e difficilmente sarebbero sufficienti per motivare la fatica dello studio del linguaggio Prolog.

Qui, al contrario, l'obiettivo è quello di stimolare il docente e la sua classe a trovare modi interessanti di *apprendere e riflettere sulla lingua* grazie ad un uso creativo di un linguaggio di programmazione. I concetti e le tecniche di programmazione, che giocoforza sono presentate ma

2 A partire dalla definizione di Jeanette M. Wing http://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

3 G.Casadei, P.Cuppini, A. Palareti, Informatica per le discipline umanistiche. Applicazioni didattiche del Prolog, Zanichelli, 1989.

non approfondite a livello di un manuale universitario, sono al servizio della didattica della lingua, e non viceversa.

La domanda che probabilmente si porrà il lettore è: ma perché è stato scelto proprio questo dominio, così lontano da computer e programmi? Nella lingua non ci sono numeri, non ci sono grandezze fisiche: cosa ci sarà mai da calcolare? E perché andare a disturbare i docenti di Italiano e Latino, che non sentono nessun bisogno di introdurre il coding nelle loro classi?

I motivi di questa scelta sono in realtà più di uno: prima di tutto, si tratta di una sfida. L'intento è quello di offrire un esempio di esplorazione di campi diversi da quello scientifico, campi in cui generalmente i docenti non hanno grandi competenze informatiche (e proprio per questo hanno bisogno di maggiori stimoli). Una mancanza di competenze derivata dalla separazione netta tra area umanistica e area scientifica che ha radici storiche profonde: la divisione medievale delle arti liberali in trivio (grammatica, retorica e dialettica) e quadrivio (aritmetica, geometria, astronomia e musica), la definizione di curricoli scolastici e universitari; ma, a mio avviso, deriva anche da una generale incomprensione della natura dell'informatica, intesa come tecnica che tratta numeri (in fondo "computer" significa "calcolatore", no?).

Informatica però significa "trattamento automatico dell'informazione", e informazione non equivale solo a "quantità numeriche". La linguistica computazionale – che si occupa di testi e letteratura – è una branca dell'informatica (o meglio, una disciplina all'incrocio di altre discipline) che ormai esiste da anni e ha una sua dignità che non deve essere più dimostrata. Esiste addirittura un'informatica umanistica,[4] che nel mondo prende il nome di "digital humanities". Meno frequente, ma altrettanto produttiva, è la direzione inversa, cioè l'applicazione di concetti e metodi delle discipline umanistiche (come la linguistica) all'informatica. In fondo esistono circa duemila linguaggi di programmazione e una biblioteca sterminata di testi scritti: possibile che a nessuno venga in mente di studiarli come tali?[5]

Il secondo motivo è legato quindi al fatto banale che se nella lingua non ci sono numeri questo non significa che non si possa utilizzare un computer in maniera interessante. Anzi: non solo si può, ma per certi versi è ancora più sensato. Se per lavorare in classe con la fisica e un computer o si usano sensori e attuatori, oppure si entra nel campo delle simulazioni (con i vantaggi e i limiti di questo approccio didattico), per lavorare con la lingua con un computer non serve niente altro. I computer sono tradizionalmente ben equipaggiati per trattare la lingua, soprattutto quella scritta, anche se oggi se la cavano bene anche con il parlato. Sanno produrre stringhe di caratteri, frasi e persino testi veri, non solo *simulazioni* di testi. Il prodotto di un automa linguistico non è un disegno o un'animazione che va interpretata come schema di un fenomeno fisico: *è il fenomeno stesso.* Questa situazione, che è simile a quella della matematica, deriva dal fatto che la lingua può essere trattata come un insieme di simboli, che possono essere codificati, analizzati, combinati, letti e prodotti. Non significa naturalmente che *tutte* le competenze linguistiche possano essere apprese in questo modo; e infatti l'oggetto delle proposte che trovate più avanti è spesso all'interno del

4 Per esempio, esiste un corso di laurea a Pisa: https://www.unipi.it/index.php/lauree/corso/10456

5 Un tentativo di lavorare in questa direzione, ormai vecchio di qualche anno, lo trovate qui: http://www.lynxlab.com/staff/steve/public/docu/lidia/

sottoinsieme della grammatica, intesa come insieme delle regole che governano la produzione di enunciati, o più in generale di sequenze di simboli.

Il terzo motivo è più sottile: la scrittura di codice sorgente è un'attività che viene di solito messa tra parentesi dagli stessi informatici, come se l'informatica fosse una scienza pura che si occupa di problemi e algoritmi che li risolvono. Invece l'informatica di tutti i giorni è una *pratica* fatta di letture, scritture, condivisioni, analisi, confronto e modifica di codici sorgenti che sono *testi;* i quali testi sono scritti in un linguaggio di programmazione che pur essendo più rigido di una qualsiasi lingua naturale non è meno flessibile e capace di adattarsi alle idiosincrasie e preferenze dell'autore. E chi, se non i docenti di lingue e letteratura, possono capire e apprezzare le genealogie dei linguaggi di programmazione, le sofisticate differenze tra le loro grammatiche, le diatribe infinite sugli stili di scrittura, le competizioni sulla qualità estetica del codice sorgente?

Torniamo però alla motivazione più didattica e cerchiamo di chiarirla con qualche esempio. Nell'attività "E perché non un'automa?"[6], l'obiettivo è costruire insieme un *automa linguistico*, cioè un programma in grado di simulare un parlante della lingua Italiana, in una situazione specifica molto semplice: premettere l'articolo ad un nome.

Perché proprio gli articoli? Perché in Italiano sono abbastanza semplici (almeno a prima vista), ma non così semplici come, ad esempio, in Inglese. Perché si prestano a discutere di regole, di eccezioni, di ambiguità; permettono di andare a toccare la storia della lingua e le parentele tra lingue romanze; hanno aspetti fonetici e aspetti semantici; eccetera.

E a che età potrebbe essere proposta questa attività? L'oggetto è costituito da conoscenze di base, che fanno parte del programma della terza classe della scuola primaria. Potrebbe darsi che i concetti utilizzati nell'attività siano troppo complessi per quell'età, almeno nella forma qui proposta. Si può però ipotizzare che si tratti di una ripresa di quelle conoscenze da parte di un gruppo di ragazzi più grandi, per esempio nella secondaria inferiore. Si può immaginare che lo scopo sia quello di costruire un programma che aiuti i bambini più piccoli, o di altre lingue madre; oppure può essere il punto di partenza per un confronto tra lingue diverse nel quadro delle attività di apprendimento della seconda lingua. Non lo so con certezza, e credo che spetti ad ogni docente trovare il momento ed il modo di introdurre un'attività nella sua classe, riducendola, modificandola, adattandola al contesto reale.

Non sono invece sicuro che il problema generale del rapporto tra grammatica e vocabolario, il tema della categorizzazione della parti del discorso e della sua importanza pratica, o le differenze tra le lingue, siano troppo difficili per poter essere affrontate anche nella scuola primaria. La grammatica probabilmente è considerata oggi una maniera "vecchia" di affrontare la lingua in termini di adeguamento ad una norma; qui, al contrario, viene presentata come una maniera intelligente di affrontare tutte le possibili varianti di una istanza comunicativa. Categorizzare dei fenomeni e scoprire una regola significa poter trattare casi diversi da un unico punto di vista. E' una competenza che sta alla base di ogni attività di riflessione sul mondo, anche al di fuori del cosiddetto "pensiero computazionale".

6 Si, l'apostrofo è voluto. Si capirà perché durante la lettura

In realtà l'oggetto è abbastanza indifferente, perché non si tratta di un'attività per insegnare *cosa sono gli articoli e come si usano*. Costruire un automa in grado di assegnare la forma giusta di articolo ad un nome è soprattutto un modo di *riflettere sulle competenze linguistiche apprese*, di scomporle, di rappresentarle in modo da essere in grado di applicarle anche a casi particolari e poi di generalizzarle.

Lo stesso tipo di approccio governa l'attività sulla coniugazione di un verbo immaginario, in Italiano o in Latino. Qui le regole sono note e già rappresentate come tabelle pronte per l'uso; ma costruire un programma in grado di applicare uno schema ad un verbo significa capire la ratio di quelle regole, i loro limiti di applicazione, essere in grado di smontare e rimontare una voce verbale verificando ogni volta che l'ipotesi fatta sia completa e corretta, o per lo meno sostenibile.

Un'altra serie di attività si svolge attorno alla costruzione di un testo vincolato, in particolare quello poetico. Per quanto abbiamo cercato di non infastidire i sacri numi, avvicinarsi alla poesia con un computer può suonare sacrilego. Però non è la prima volta che succede: si pensi agli esperimenti di Nanni Balestrini negli anni 60. Nell'introduzione al suo Tristano[7] (ripubblicato nel 2007 da Derive e Approdi) scrive:

> *Nel 1961 avevo composto Tape Mark I, un esperimento poetico realizzato sfruttando le possibilità combinatorie di un calcolatore elettronico IBM (così allora veniva chiamato il computer). Una serie di spezzoni di frasi venivano montate in successione, fino a formare sequenze di versi, seguendo semplici regole trasformate in un algoritmo che guidava il lavoro della macchina. Il numero dei risultati possibili era grandissimo, e una piccola serie di varianti, sufficiente a mostrare il senso dell'operazione, venne pubblicata sull'Almanacco Bompiani 1962.*

Più recentemente,[8] è assurto agli onori della cronaca un chatbot (cioè un programma che è in grado di interagire testualmente con umani, per esempio tramite Twitter) sviluppato da Li Di, nel centro di ricerca Microsoft in Cina. Il progetto di ricerca aveva come obiettivo quello di creare un chatbot "simpatico", con cui fosse divertente interagire, e questa capacità Xiaoice ha dovuto apprenderla dall'esperienza: si è nutrito di milioni, o miliardi, di interazioni con utenti reali. I ricercatori cinesi hanno avuto ad un certo punto l'idea di dare in pasto al programma una quantità notevole di poesie moderne cinesi. A questo punto, Xiaoice è stato in grado di comporre circa diecimila poesie originali. Alcune delle quali sono state selezionate e pubblicate da una casa editrice cinese, Cheers Publishing. Ad esempio questa:

> *Through the blur of tears, nothing is clear*
>
> *My life is art;*
>
> *Drifting clouds at dusk in the western sky,*

7 http://www.deriveapprodi.org/2007/11/tristano/

8 Alcune "poesie" di Xiaoice sono tradotte in italiano nel quadro dell'articolo dedicato al tema da Sergio Basso, La lettura, n° 292 del 2 luglio 2017, pag. 23; letto su https://www.pressreader.com/italy/la-lettura/20170702/281771334211138

With my broken palms I pray.

con le reazioni scandalizzate degli addetti ai lavori che potete immaginare.

Ma in generale un po' di leggerezza nella trattazione della poesia forse gioverebbe al suo insegnamento. Recentemente ho riletto le poesie di Fosco Maraini, come questa:[9]

Il lonfo non vaterca né gluisce

e molto raramente barigatta,

ma quando soffia il bego a bisce bisce

sdilenca un poco, e gnagio s'archipatta.

Come considerarle? Un attentato alla lingua? Un divertimento folle? Per quel che ci interessa qui, potrebbero essere dei modelli di costruzione di testi in cui in una struttura metricamente ineccepibile vengono inseriti dei lemmi di una lingua inesistente. Ma sono parole che *potrebbero* essere italiane, anche se non lo sono. Anzi, per alcune sembra quasi di intuire un significato, per analogia con altre parole o perché sembrano onomatopee. Si può inventare una lingua che assomigli all'italiano? Quanto ci si può allontanare dal modello reale perché suoni ancora italiano? E come si procede? Come faceva Dario Fo (o Carlo Bonomi, o Charlie Chaplin) a costruire il suo grammelot?

Sono domande che preludono ad ogni possibile costruzione automatica di un testo. Domande che toccano l'accettabilità delle parole, le competenze fonetiche dei parlanti, la distribuzione delle consonanti. Anche questo è, a mio parere, un modo di acquisire consapevolezza della lingua. Ed è su questo piano di interrogazione continua che le attività di "coding" possono avere senso.

C'è infine un'altra serie di attività che travalicano i confini della lingua e spaziano in altre aree della produzione umana, da quella visiva a quella musicale, passando per i giochi. Ciò che unisce tutte queste attività, e le giustifica nel contesto di un progetto formativo, è l'approccio: si tratta, come si vedrà, di costruire grammatiche che permettano di *tradurre*, cioè di trasformare sequenze di simboli in altre sequenze che possono avere una rappresentazione non linguistica: un'immagine, un brano musicale, un testo Braille, una strategia di gioco. E questo approccio, al di là delle diverse definizioni di "computazionale" che si trovano in circolazione, è probabilmente il cuore stesso di tutta l'informatica. Inserire queste attività in un testo dedicato alla lingua è soprattutto un modesto tentativo di invitare il docente a non fermarsi nel recinto dell'insegnamento della lingua scritta, ma a cercare un denominatore comune a tante attività produttive e a stimolare gli studenti a esplorarle tutte in maniera creativa e, insieme, rigorosa.

1.3 Il modello didattico

Probabilmente i docenti di materie umanistiche che leggeranno questo testo troveranno imprecisioni, errori, mancanze. Mea culpa: non sono un linguista, né un docente di lingua. Tutte le

9 Gnosi delle Fanfole, Dalai Editore, 1994. Purtroppo introvabile. Un blog ricchissimo di esempi di poesie che giocano con il linguaggio in maniera forte è https://antoniobux.wordpress.com/

attività si possono senz'altro correggere e migliorare, o ripensare da zero. In generale, quanto scritto rappresenta solo un esempio di *una possibile modalità* di utilizzo della programmazione come strumento di costruzione collettiva di conoscenze, da adattare (o ripensare) nei contesti specifici della classe.

Allo stesso tempo i docenti ferrati in informatica troveranno che i concetti cardine della programmazione sono banalizzati e che gli esempi scelti sono piuttosto semplici (almeno in apparenza, ma si vedrà che ci sono spazi per l'approfondimento anche nel progetto più banale). Sono stati costruiti in questo modo per far emergere meglio una strategia didattica, che è basata non soltanto sulla esibizione di soluzioni, ma soprattutto sulla *discussione* che fa emergere ipotesi, sulla *sperimentazione* diretta, sul *miglioramento* incrementale, sulla proposta di *sviluppi*.

Non è un caso che gli esempi proposti non siano quelli di un quiz o di un programma che "interroga" lo studente (come in tanti esempi che invece si possono trovare in rete[10]). Anzi: il modello è esplicitamente un altro, quello dell'insegnamento *al* computer *da parte* dello studente, o meglio da parte di un gruppo di studenti. Insegnamento di cosa? Di conoscenze e competenze che, almeno in parte, fanno parte del bagaglio di conoscenza implicita che può essere resa esplicita grazie proprio all'attività proposta, e in parte devono essere raccolte. Naturalmente è possibile che non tutti conoscano esattamente, ad esempio, la lista delle condizioni di applicazione di una forma di articolo (in particolare, i ragazzi di lingua madre diversa dall'Italiano): ma è parte integrante dell'attività proposta la ricerca di queste informazioni, attraverso l'intervista a utenti più esperti (ragazzi più grandi, genitori, docenti), la lettura di testo scolastico di grammatica Italiana, di un'enciclopedia[11] o di Wikipedia[12] – che rappresenta la sfida più difficile. Un'attività che non si svolge tutta davanti ad un computer, né è un'attività in solitaria: il gruppo è fondamentale per la discussione, per la ricerca, per la costruzione. Si possono anche organizzare due gruppi che si scambiano i ruoli: un gruppo scrive il programma, l'altro lo testa per vedere se si comporta "bene", cioè secondo le aspettative.

Costruzione di conoscenza significa *ricerca*, non trasmissione. Il punto non è come trasmettere conoscenze agli studenti; il punto è come un gruppo di studenti arriva a costruire una conoscenza stabile, condivisibile, rappresentata in una forma pubblica, a partire da elementi sparsi, non strutturati, contraddittori o apparentemente tali. E' possibile che per alcuni studenti questa sia *anche* un'occasione per imparare da zero delle conoscenze di base, oltre che per costruire su quella base un modello funzionante; ma non è l'obiettivo primario, perché questa attività non si sostituisce ad una lezione di grammatica. Per intenderci: se si dovesse partire con lo *spiegare* che esistono articoli determinativi e indeterminativi, l'attività sarebbe troppo lunga e complessa e di dubbia efficacia. Qui si suppone semplicemente che il concetto sia almeno vagamente conosciuto, ma forse non utilizzato consapevolmente e al meglio; e che, viceversa, imparare a memoria un elenco di parole o saper rispondere ad una domanda non valga quanto la costruzione di una teoria esplicita su come si utilizza un pezzetto della lingua.

E il docente? E' importante? Qui immaginiamo che il docente, anzi i docenti, siano *fondamentali*, a differenza di quanto capita in certi modelli di "coding" presenti sul web. Il ruolo del docente non è

10 Per esempio, http://www.baby-flash.com/lavagna/articoli_sbagliati1.swf oppure http://www.softwaredidatticofree.it/schedaapostrofo1.htm

11 http://www.treccani.it/enciclopedia/articoli-indeterminativi_(La-grammatica-italiana)/

12 https://it.wikipedia.org/wiki/Articolo_(linguistica)#Articoli_indeterminativi

quello del trasmettitore di conoscenze o della guardia che assicura l'ordine in classe mentre i ragazzi giocano al computer. Il docente è la guida, che apre la discussione, aiuta a vagliare le ipotesi, suggerisce parallelismi e approfondimenti. Gli esempi di codice sorgente forniti nel testo non vanno presentati subito nella forma finale, ma vanno *costruiti* insieme ai ragazzi, un pezzetto alla volta, discutendone la scrittura, provando versioni diverse, analizzando gli errori. La sezione "Discussione" che segue ogni esempio di codice ha lo scopo di suggerire alcune piste che possono o meno essere seguite dal docente in base alla valutazione del contesto

Il docente deve, naturalmente, avere un'ampia competenza nell'area della lingua (delle lingue), della glottodidattica e della letteratura; ma per far funzionare la macchina digitale deve essere in grado di padroneggiarne la lingua. Deve essere in grado di proporre delle tecniche alternative, di spiegarne i vantaggi e gli svantaggi - e da questo punto di vista può essere utile che a gestire l'attività sia una coppia di docenti, di cui almeno uno con competenze informatiche più avanzate, che sia un animatore digitale o un docenti di matematica.

Una nota sul concetto di "problema", che nelle proposte didattiche di coding è spesso presentato come il punto di partenza da cui discende la costruzione della soluzione algoritmica: qui non si parte da un *problema*, ma da un *progetto* di costruzione di qualcosa. Strada facendo, appaiono dei problemi (di lingua, o di linguaggio) che vanno affrontati, evitati o risolti. Alcuni si riveleranno alla fine problemi irrisolvibili, almeno con le risorse a disposizione, e anche questa è una scoperta importante: l'informatica, a differenza della matematica e della logica, ha dei limiti fisici che le sono dati dal contesto in cui opera.

1.4 La lezione di Papert

L'approccio qui seguito deve molto al lavoro di Seymour Papert, o per lo meno a quello che ne ho capito io. Ho scoperto Papert, il Logo e tutta la letteratura pedagogica connessa, alla fine degli anni '80, quando insegnavo (piuttosto sperimentalmente) informatica in una scuola privata di Roma. Come me, decine di docenti negli stessi anni andavano disegnando girandole ricorsive, un po' come ora si fa con Scratch. A me non interessava particolarmente la geometria, innovativa o meno, né l'educazione artistica. Il mio interesse era, allora come adesso, per la metafora della programmazione come "insegnamento al computer", che conteneva tanti aspetti davvero nuovi (almeno per me).

Prima di tutto, rovesciava il rapporto tra bambino e ambiente educativo. Usando uno strumento che per l'epoca era senz'altro da grandi, al bambino veniva proposto di insegnare ad un "tu" disponibile e infaticabile. Che significa insegnare? Si può insegnare qualcosa, la si può spiegare, solo se la si capisce bene. Insegnare diventa un modo di capire meglio, di approfondire, di superare le approssimazioni e definire i concetti. C'era, insomma, un valore forte legato all'apprendimento di una maniera di pensare. Un valore che farei fatica, oggi, a far coincidere con l'insegnamento del "pensiero computazionale", perché non aveva a che fare con sequenze e scelte, ma con la più generale consapevolezza nell'uso dei concetti e delle loro relazioni.

In secondo luogo, la tipica interazione raccontata da Papert non era bambino-computer, ma bambini-computer. Parlare a voce alta, presentare delle ipotesi e discuterle, poi provarle e verificare il risultato, mi sembrava un modo di affrontare qualsiasi progetto in maniera molto più efficace della colluttazione solitaria con un libro. E questo detto da qualcuno che come me ha un percorso formativo tutto classico, dentro ai libri e le biblioteche. Certo non si lavoro in gruppo solo al computer; ma sottolineare che questo tipo di lavoro può essere efficace solo se viene fatto in gruppo mi pareva un grande passo avanti.

Infine mi colpiva il valore epistemologico dell'approccio di Papert. Affrontare la geometria in maniera costruttivista (o costruzionista, per essere fedeli terminologicamente) non significa inventare ogni volta l'acqua calda: significa cercare delle regolarità, generalizzarle, riapplicarle in altri casi, cercare i casi limite. Questo modo di apprendere, così simile a quello degli adulti che costruiscono teorie (poco importa se in campo umanistico o scientifico) mi pareva molto più efficace della semplice trasmissione di informazioni.

Insomma il computer programmabile si rivelava un ambiente educativo flessibile, modificabile, altamente interattivo, dove fare pratica scientifica collettiva. Ma allora perché usarlo solo per la geometria? E infatti, ci sono report di lavori creativi svolti da bambini utilizzando il Logo che avevano per oggetto la creazione di frasi casuali a partire da una struttura, o di coniugazioni in altre lingue.[13]

1.5 Strutture linguistiche e creatività

L’incontro tra informatica e letteratura avviene ufficialmente, almeno in Italia, quando nel 1949 Padre Busa SJ si dedica all’immane compito di compilare un Index Thomisticus, cioè un repertorio di tutti i termini utilizzati dall’Aquinate nelle sue opere. Per farlo, chiede il supporto dell’IBM (parlando con il suo fondatore, Watson) e inizia un lavoro di lemmatizzazione durato trent’anni. Dopo la versione cartacea (1980) e quella su cdrom (1989), nel 2005 nasce la versione web.[14]

La linguistica computazionale si presenta così con un aria seriosa, doppiamente sostenuta dall’oggetto (il testo classico) e lo strumento (il programma di lemmatizzazione e ricerca), per non parlare dello scopo scientifico.

Ma ci sono stati altri incontri meno nobili, come quello tra il libro cartaceo Cent Mille Miliards de Poèms di Raymond Queneau e il web. Per chi non avesse avuto la fortuna di sfogliare quel meraviglioso oggetto, si tratta di un libro pubblicato nel 1961 che raccoglie dieci sonetti di quattordici versi ognuno. La peculiarità che lo rende unico è la pagina è tagliata in orizzontale in modo da rendere ogni verso un oggetto autonomo; è possibile così leggere (e costruire con la mente) un sonetto costituito, poniamo, dal primo verso della prima pagina, il secondo dalla decima, il terzo dalla quinta, e così via. Le possibilità totali sono 1014, cioè appunto 100.000.000.000.000.[15]

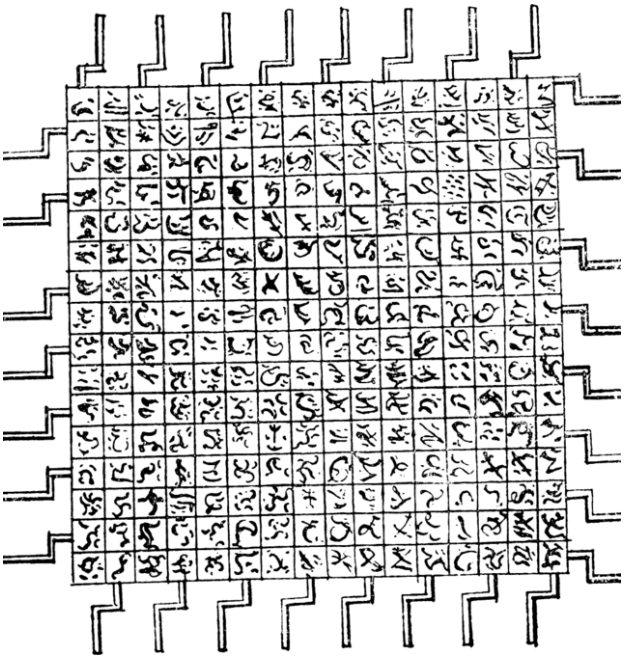
13 Una serie di questi lavori sono raccontati al capitolo 8 di quest'articolo di Papert del 1970: http://www.citejournal.org/volume-5/issue-3-05/seminal-articles/teaching-children-thinking/

14 http://www.corpusthomisticum.org/it/index.age

15 Di questa macchina per generare sonetti ne esistono varie versioni consultabili su web, come per esempio questa: http://www.growndodo.com/wordplay/oulipo/10%5e14sonnets.html

Queneau realizza (cioè “dimostra la possibilità”) di qualcosa che nel cielo delle invenzioni letterarie era ben nota. A partire per lo meno dalla macchina creata dagli scienziati dell’Accademia di Laputa:

> La superficie risultava di vari pezzetti di legno, grossi press’a poco come dadi, alcuni di maggiore dimensione degli altri. Erano tutti congiunti da esili fili di ferro. Incollata sopra le quattro facce dei pezzetti di legno era della carta, e su questa si trovavano scritte tutte le parole della loro lingua, coniugate nei diversi modi e tempi e declinate nei vari casi, ma senza ordine veruno. Il professore m’invitò a prestare attenzione, ché appunto s’accingeva a mettere in moto la macchina. Ciascun discepolo prese, al cenno del maestro, un manico di ferro (ce n’erano quaranta fissati intorno agli orli della macchina) e d’un tratto lo fece girare. Naturalmente la disposizione delle parole cambiò in tutto e per tutto. Il maestro ordinò allora a trentasei scolari di leggere pian pianino i vari righi così come apparivano sulla macchina; e quando quelli trovavano tre o quattro parole unite insieme che potevano far parte d’una sentenza, le dettavano ai quattro rimanenti discepoli che fungevano da scrivani (Jonathan Swift, I viaggi di Gulliver, Traduzione di Carlo Formichi, a cura di Masolino d’Amico, Mondadori, Milano, 1982, p. 393).



E così di seguito, passando per Borges, Levi, Landolfi e Dahl. Molti altri esempi sono citati nella trascrizione di una bellissima conferenza del 2015 tenuta da Paolo Albani[16] (a meno che non sia anche questo un testo generato automaticamente).

Cosa mostra davvero questo strano artefatto, nella versione cartacea come in quella digitale? Che la letteratura (e in particolare la poesia) non è tutta intuizione ed espressione libera. Che il gioco tra sistema e creatività, tra regola ed eccezione, non è proprio così chiuso come sembra. La poesia, in particolare, nasce proprio dal vincolo (tematico, formale), come orizzonte e come sfida. Non lo dico io, lo dice Calvino: la letteratura è

> “un’ostinata serie di tentativi di far stare una parola dietro l’altra seguendo certe regole definite, o più spesso regole non definite né definibili ma estrapolabili da una serie di esempi o protocolli, o regole che ci siamo inventate per l’occasione cioè che abbiamo derivato da altre regole seguite da altri” (Cibernetica e fantasmi. Appunti sulla narrativa come processo combinatorio, in: Una pietra sopra. Discorsi di letteratura e società, Einaudi, Torino, 1980, pp. 164).

16 http://www.paoloalbani.it/Letteraturacombinatoria.pdf

E cosa fa il poeta quando crea, se non andare a pescare nella sua memoria linguistica e scegliere combinazioni di parole, vincolate da regole precise (come il metro o la rima)? Certo, la scelta è anche governata dal significato – anche se in maniera difficile da precisare. Il poeta parte con l'idea da esprimere e cerca le parole più adatte? Oppure si lascia guidare dalle parole stesse, sfruttando somiglianze fonetiche, rimandi per analogia o opposizioni? O ancora, più probabilmente, attua un misto delle due strategie? Insomma: come si scrive, *praticamente*, una poesia?

D'altra parte si sente spesso parlare dell'introduzione del coding a scuola come modo per stimolare la creatività, la quale creatività starebbe nella possibilità di costruire storie con personaggi animati che si muovono sullo schermo e dicono qualcosa. Questa è creatività? O la creatività starebbe nel risolvere una serie predefinita di problemi, magari scegliendo la soluzione corretta prevista dall'autore?

Personalmente penso che la creatività sia qualcosa di più complesso, che non ha necessariamente a che fare con i colori e i suoni, e sicuramente più con l'invenzione di problemi che con la loro soluzione applicando un algoritmo. La creatività è quella che permette di improvvisare su un tema musicale. Come si fa? Certo non pestando tasti a caso. Improvvisare significa avere in testa il tema, la sua struttura, provare a muoversi all'interno di uno spazio armonico, seguire un'idea e verificare se funziona. Il tutto in tempo reale, il che lo rende maledettamente difficile. E difatti l'improvvisazione si studia.[17]

Di qui l'idea di proporre delle attività didattiche di coding intorno ai temi della forma e della variazione, delle categorie e del caso, dell'accettabilità delle varianti. In un periodo in cui il *machine learning* sembra riproporre il vecchio mito dell'intelligenza artificiale, viene voglia di ragionare intorno ai processi creativi anche utilizzando paradossi, e di provare a costruire in classe un automa che sia in grado (se non proprio di scrivere poesie originali, o di produre variazioni per sottrazione da una poesia di Walt Whitman[18]) almeno di inventare ricette gastronomiche sempre nuove, che tutto sommato sono sempre forme di testo vincolate.[19]

Per restare nel dominio letterario, si possono consultare due modesti esempi - creati dal sottoscritto - di macchine figlie di quella di Queneau (ma che pescano nel testo di due classici sempreverdi come l'Inferno di Dante Alighieri e l'Orlando Furioso di Ludovico Ariosto).[20] Oltre a Queneau, questi due oggetti digitali si ispirano più precisamente a "Il centunesimo canto. Philologica dantesca"di Luca Chiti, che è un meraviglioso esempio di centone umoristico che "crea" un intero canto giustapponendo versi esistenti ma dandogli un senso completamente nuovo. Per realizzarli, ho dovuto affrontare problemi letterari, come la definizione di rima, di struttura metrica, di novità e ripetizione (oltre che qualche problema informatico, come il loop infinito o la conversione dei caratteri in UTF-8). Non ho seguito alla lettera le indicazioni di Nanni Balestrini, ma ci sono andato vicino. Ed ecco apparire terzine più o meno improbabili come la seguente:

17 Su questo tema ho trovato ispirante, anche se non condivisibile nei commenti sulla tecnologia, questo saggio su creatività e apprendimento delle lingue di Ilaria Pezzola, http://www.itals.it/il-ruolo-della-creativit%C3%A0-nell%E2%80%99apprendimento-linguistico-teorie-e-applicazioni

18 https://scratch.mit.edu/projects/12331423/

19 http://www.lynxlab.com/staff/steve/public/ricette

20 Li trovate qui http://www.lynxlab.com/staff/steve/public/inferno e qui http://www.lynxlab.com/staff/steve/public/orlando

Parlando cose che ‘l tacere è bello

rispuosemi: «non omo, omo già fui

venimmo al piè d’un nobile castello

o come quest'altra:

Nel nome che sonò la voce sola

poscia vid’io mille visi cagnazzi

cosí vidi adunar la bella scola

Una valutazione estetica del risultato? Non è l’obiettivo, anche se può essere divertente provare e riprovare, fino a far emergere dei frammenti di senso che possono essere anche divertenti. Ma quelle che mi paiono importanti, come sempre, sono le domande che emergono ogni volta che si prova a realizzare un modello funzionante di una teoria: come si riconosce una rima? Come si produce una struttura metrica? Come si ottiene un testo sempre diverso, ma che abbia almeno la parvenza di un significato? E viceversa, come si riconosce che il testo *non* è stato scritto da un poeta umano?

Mi sembrano tutte domande legittime da porsi in una classe che studia lingua o letteratura: portano con sé riflessioni e discussioni che integrano, anche se non sostituiscono, l’apprendimento di nomi di forme di testo come “trimetro scazonte” o “endecasillabo sciolto” o di opere particolari.