Ampala Alessandro
Borrelli Mattia

Relazione Esercizio 2

La sperimentazione dell'applicazione Es2App, il cui funzionamento si basa sull'utilizzo di un algoritmo per il calcolo della distanza di edit tra due stringhe facente uso di memoization, ha fatto emergere considerazioni interessanti.

L'obiettivo del programma, cioè elencare per ogni parola contenuta in un file (detto *correct*) la lista di parole, contenute in un secondo file (il *dictionary*), con distanza di edit minima da questa, è stato raggiunto in un tempo pari 3.38.85 minuti. Un risultato da considerarsi più che accettabile considerando la presenza di 49 e 661562 parole nei file *correct* e *dictionary* presi in esame. L'analisi di una singola parola impiega approssimativamente sette secondi, anche se si tratta di una media fortemente sbilanciata da valori estremali, causata dall'implementazione dell'applicazione. Infatti, oltre alla complessità dell'algoritmo di calcolo dell'edit distance occorre tenere anche in considerazione il numero di confronti effettuato su una parola, corrispondente anche al numero di chiamate dell'algoritmo stesso. Concretamente, l'applicazione esegue una sola passata del dictionary, aggiungendo ad una lista ogni parola che abbia edit distance pari all'attuale minimo. Qualora sia trovata una parola con distanza di edit minore dell'attuale, la lista viene svuotata e reinizializzata con quest'ultima parola. Inoltre, la ricerca termina immediatamente nel caso sia individuata un'edit distance pari a 0, poiché ciò significa le due parole confrontate sono uguali, e dunque non sarà possibile trovare un'altra parola con edit distance minore a meno di ripetizioni nel file *dictionary*. Ai fini dell'esperimento consideriamo una parola la cui edit distance minima è 0 come una parola "corretta". Alla luce di questo, è chiaro come il tempo che l'applicazione impiega nel completare la ricerca per un parola di *correct* sia influenzato molto anche dalla posizione, se presente, della parola analizzata nel file *dictionary*. Comunque, l'efficienza dell'applicazione non è nemmeno lontanamente paragonabile ad una simile che utilizzi una versione dell'algoritmo implementata senza l'uso della programmazione dinamica.

Considerando l'output come un elenco delle liste di possibili correzioni di ciascuna parola, sbagliata a causa di un errore di battitura, ritroviamo alcuni risultati insoddisfacenti. Infatti, osservando il prodotto del programma è possibile notare come solo per alcune parole errate sia presente nella lista la correzione. Questi risultati sono da imputare all'impossibilità di determinare il contesto nel quale viene utilizzata la parola, pertanto indipendenti dall'algoritmo. Infatti, queste mancanze si presentano unicamente in due casistiche:

> **Parola da considerarsi errata nel contesto ma presente in *dictionary*.**
> Ad esempio la parola "selice" è da intendersi contestualmente come "felice", ma data la presenza di "selice" nel file *dictionary* la parola in esame è da considerarsi corretta.

Ampala Alessandro
Borrelli Mattia

Parola la cui versione corretta nel contesto ha un edit distance diversa dalla minima.
Ad esempio “squola”, versione ortograficamente scorretta di “scuola”, ha un edit distance minima pari ad 1 ed una lista contenente la parola “suola”, contestualmente scorretta.

Per un’applicazione più specifica, sarebbe possibile eliminare quasi del tutto il secondo caso modificando l’algoritmo in modo tale da aggiungere alle operazioni possibili la sostituzione di una lettera (e non considerare quest’ultima come combinazione di cancellazione + inserimento). Una soluzione simile risolverebbe il problema, ma al costo di rendere l’algoritmo più complesso e le liste di parole idonee generalmente più lunghe.