Ho fatto una bozza di report (poi da trasferire su Power Point). Ditemi se l’impronta vi piace.

Dove c’è la scia di punti interrogativi bisogna ancora scrivere.

REPORT

PROGETTO: MISSPELLING

INTRODUZIONE AL PROBLEMA

Gli odierni strumenti tecnologici quali smartphone, tablet e computer sono dotati di una serie di programmi e software che controllano l’ortografia delle parole digitate, correggendo automaticamente il termine scorretto o proponendo, in fase di digitazione, una serie di dizioni che più somigliano alle potenziali parole a cui si può riferire la sequenza di lettere già battute a tastiera.

Il progetto da noi scelto (Misspelling) ha riproposto questo genere di problema, pertanto abbiamo scelto di provare a implementare anche noi un correttore automatico di parole.

IL PROGETTO

ORGANIZZAZIONE

Per semplicità grammaticale e lessicale abbiamo scelto la lingua inglese in quanto scarsa o priva di determinati simboli (“’”, accenti) e generi (maschile / femminile).

Inoltre abbiamo effettuato un’ulteriore scrematura non considerando simboli di punteggiatura e caratteri speciali quali “#”, “-“, “@”..., ma considerando solamente lettere e numeri.

La piccola modifica che abbiamo effettuato rispetto alla specifica, è il fatto di aver preso in considerazione un libro in inglese “THE FELLOWSHIP OF THE RING” invece che i Tweet del famoso Social Network.

Quindi abbiamo sviluppato il progetto in linguaggio Python vs 3.6 e abbiamo aggiunto un ulteriore metodo di codifica; oltre alla normale dicitura ed errori che possono incorrere digitando su una tastiera qwerty, abbiamo anche preso in considerazione i problemi che possono sorgere utilizzando una tastiera Braille.

MODELLO

Per risolvere questo tipo di problema, abbiamo utilizzato un Hidden Markov Model, una struttura dinamica che calcoli, per ogni parola, una serie di parametri.

- Le lettere (su tastiera qwerty e Braille) adiacenti.

Per la tastiera qwerty abbiamo considerato adiacenti le lettere vicine. Per esempio, le lettere

adiacenti alla “a” sono “q”, “w”, “s”, “z”.

Per la tastiera Braille, invece, abbiamo considerato adiacenti le lettere che differiscono l’una

dall’altra per la presenza o meno di un puntino. Per esempio, abbiamo considerato adiacenti alla

“d” i caratteri “g”, “n”, “4”, “c”, “e”.

|  |  |
| --- | --- |
| d g | d g |
| d n  d n |  |
| d 4 | d 4 |

- Codifica ??????????

- La classe PriorGenerator ha al suo interno dei metodi per

a...) eliminare i caratteri speciali che non consideriamo quali (“#”, “@”, “-“), accenti e punteggiatura.

b) contare il numero di occorrenze delle lettere

c) aggiungere le parole trovate nel testo al dizionario

d) calcolare la frequenza di certe lettere avendo già calcolato la frequenza delle lettere precedenti ad esse

e) scrivere in un file le parole del dizionario trovate.

- Nella classe Hyperviterbi vi sono dei metodi per:

a) calcolare la distanza di Levenshtein tra tutte le parole del dizionario e la parola data, considerando i tre

tipi di errore (inversione, sostituzione e omissione).

b) quando si analizza una intera frase il programma la suddivide nelle sue parole e per ciascuna calcola

la sequenza di caratteri più probabile

c) una volta scorse tutte le lettere di una parola e calcolato il predecessore più probabile, Viterbi calcola lo

stato finale più probabile e quindi la parola.

- il Main carica i simboli che non consideriamo, il dataset ed esegue il correttore ortografico.

- risultati (numerici, grafici) indicatori di bontà ???????????????

- conclusione: come può funzionare meglio, se non ha funzionato perché. ???????????????