REPORT

Giulia Boifava 781250

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 come dataset un libro di narrativa in lingua inglese, 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.

L’interfaccia è stata sviluppata in HTML. Una volta digitata la parola appare un menù con una lista di 6 alternative, scelte dal programma in base all’adiacenza di eventuali lettere scritte per errore.

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 |

- 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

Il correttore è in grado di riconoscere e correggere gli errori di battitura di una parola, ma non è in grado di riconoscere gli errori di semantica.

Digitata una parola, l’interfaccia HTML propone una lista di sei possibili termini in base alla distanza di Levenshtein.

Esempio:

TASTIERA QWERTY

Parola digitata: “hello”

Suggerimenti: “hell”, “he’ll”, “hells”, “tell”

TASTIERA BRAILLE

Parola digitata: “point”

Suggerimenti: “print”, “points”, “prints”

Ogni 10 errori inseriti il margine di correttezza è dell’82%.

Con una perturbazione del 20% la stima di correttezza scende al 71%.

Se il testo non contiene errori il test dà l’85% di correttezza.

Tuttavia la stima di correttezza è migliore sulle frasi corte rispetto alle frasi più lunghe.

Non corregge invece gli errori di semantica: la parola successiva non viene calcolata in base alle parole precedenti, quindi il senso della frase è totalmente ignorato.

CONCLUSIONI

I maggiori problemi sono il fatto che non viene distinto il singolare dal plurale delle parole, quindi anche se un termine è scritto correttamente, viene considerato errato per il fatto che abbia una lettera in più o in meno.

“terms” viene corretto con “term”. “not” viene corretto con “no”.

In frasi più corte la stima dicorrettezza è comunque migliore rispetto alle frasi più lunghe che contengono subordinate e / o coordinate.

Il codice può essere migliorato applicando un’analisi di correttezza anche sulle parole precedenti e successive, in modo da avere un senso generale della frase e cercare di intuire alemno i generi delle parole.