Metodi Probabilistici per le Decisioni
Mispelling

Andrea Angiolillo Matr. 761678
Corrado Ballabio Matr. 765244

June 27, 2016

Introduzione

All'interno degli smartphone di nuova generazione sono disponibili strumenti automatici in grado di correggere i testi scritti all'interno dei messaggi testuali che ogni giorno vengono inviati (mediante *Whatsapp*) e/o diffusi sui social network (tramite *Facebook* e *Twitter*).
In questo progetto è stata realizzata una nuova tastiera sperimentale che sfrutta le potenzialità degli *Hidden Markov Models* per correggere errori e rendere la digitazione e predizione delle parole piú precisa possibile.

Obiettivi del Progetto

- Creazione del Dataset di training e testing:
  1. Raccogliere un insieme di tweet utilizzando le API di *Twitter*
  2. Perturbare i testi introducendo il 10% di writing mistakes.
  3. Dividere i testi in training (80%) e testing (20%).
- Modello HMM
  1. Definizione della struttura *HMM* per inferire il testo corretto, date le osservazioni derivanti dalla tastiera
  2. Stima dei parametri

- Correzione su nuovi testi
    1. Inferire il testo corretto, dato il messaggio inserito dall'utente
- Analisi dei dati
    1. Stimare le capacità predittive del modello rispetto alla *ground truth*
    2. Valutare le capacità predittive del modello al crescere dei dati di training

Assunzioni

- **Scelte linguistiche**
  Per quanto riguarda la scelta relativa alla lingua utilizzata nel progetto, si è deciso di utilizzare l'inglese. Questo poichè grammaticalmente più semplice dell'italiano ma soprattutto grazie al fatto che non comprende molti caratteri di difficile gestione come le lettere accentate, che avrebbero generato troppe sfumature nella implementazione del modello e portato alla creazione di una quantità di stati decisamente eccessiva per i nostri fini.
  Per la creazione del *ground truth* vengono quindi considerati solo i simboli alfabetici (a-z) che formano gli idiomi della lingua inglese, tutti gli altri segni di punteggiatura verranno scartati in quanto non influenti nel processo di correzione.

- **Scelta delle fonti**
  La scelta dei profili *Twitter* dai quali scaricare i tweets su cui poi creare le distribuzioni probabilistiche non è stata casuale. Si è infatti selezionato un insieme di profili che si è ritenuto utilizzassero e scrivessero in un linguaggio corretto, senza uso di abbreviazioni o termini appartenenti allo slang.
  Inoltre si è preferito evitare testate giornalistiche o di divulgazione scientifica, in quanto trattando di questioni internazionali sarebbero stati presenti molti nomi propri di diverse nazionalità, termini di qualunque lingua e un uso frequente di abbreviazioni o di gergo appartenenti a specifici campi. Tutti questi elementi avrebbero portato a una "distorsione" nei valori di probabilità di transizione che non riflettono la realtà della lingua inglese.

Le fonti selezionate risultano quindi:

- *@UKLabour* il partito laburista inglese
- *@Conservatives* partito conservatore inglese
- *@David_Cameron* attuale primo ministro del Regno Unito
- *@MayorofLondon* sindaco di Londra
- *@UniofOxford* università di Oxford
- *@Cambridge_Uni* università di Cambridge

In totale risultano approssimativamente 12000 tweets.

Scelte Progettuali

Durante il progetto sono state intraprese diverse scelte progettuali, in questa sezione andremo a parlare delle motivazioni che ci hanno spinto a utilizzare specifici linguaggi di programmazione e librerie.

- Linguaggio di Programmazione - *Python*
  I requisiti che abbiamo tenuto maggiormente in considerazione per effettuare la scelta del linguaggio di programmazzione da utilizzare sono:

  1. il linguaggio deve disporre di numerose librerie scientifiche
  2. l'ambiente deve essere completamente gratuito

  Dopo numerose ricerche abbiamo scelto *Python*: un linguaggio completamente gratuito sviluppato in ANSI C, caratteristica che lo rende utilizzabile su qualsiasi piattaforma.
  Il motivo principale che ci ha convinto a scegliere *Python* è che risulta essere uno dei linguaggi con più librerie e quindi riesce ad adattarsi a qualsiasi problema in modo completamente ottimizzato, inoltre, essendo molto utilizzato, si avvale di una community molto grande.
  Una delle più valide alternative a *Python* è *Matlab*, linguaggio ottimizzato per problemi scientifici e di progettazione, ma il fatto di essere completamente a pagamento ci ha convinti a scartarlo come alternativa

- Libreria *HMM - Pomegranate*
  Vista l'incompatibilità della libreria *GHMM* per i sistemi operativi *Windows*, abbiamo deciso di utilizzare una alternativa molto valida: *Pomegranate*.

Pomegranate è una libreria per modelli grafici e statistiche bayesiane per *Python*, attualmente supporta:

  - Probability Distributions
  - General Mixture Models
  - Hidden Markov Models
  - Naive Bayes
  - Markov Chains
  - Discrete Bayesian Networks
  - Factor Graphs
  - Finite State Machines

- Approccio al progetto
  Si è deciso che per ogni lettera il set dei suoi errori possibili coincidesse con l'insieme delle lettere che sono a lei limitrofe nella disposizione della tastiera.
  Per la vera e propria fase di correzione si è deciso di effettuare l'inferenza sul modello di Markov tramite l'algoritmo di Viterbi. In questo modo sfruttando le distribuzioni successivamente fornite, è possibile ricavare la sequenza più probabile (e quindi presumibilmente corretta) di lettere che formano una parola, data l'osservazione di una stringa contenente errori.

Implementazione

- **Scaricamento tweets**
  La prima parte nell'implementazione del progetto è stata quella dello scaricamento e del salvataggio dei tweets in un unico file formato CSV, tramite la libreria *tweepy*. Successivamente questo grande documento è stato sottoposto a una serie di espressioni regolari, il cui compito era quello di "depurare" il testo dei tweets. Sono stati infatti eliminati tutti i caratteri di punteggiatura, i numeri, gli hastags, i tags, i retweets e in generale tutti i simboli non alfabetici. Il risultato finale è un file denominato *clean_tweets.csv* contenente semplice testo in inglese, pronto per essere analizzato.
  Come ultimo punto di questa fase, tramite un generatore di numeri randomici, il file *clean_tweets* è stato diviso in due diversi file. Il primo, contenente l'80% dei tweets originali è stato chiamato *gt_tweets.csv* ed è poi stato utilizzato come training set da cui ricavare informazioni sulla

distribuzione dei caratteri. Il secondo è stato denominato *lp_tweets.csv* e contiene la restante quantità del testo, funge da testing set e verrà utilizzato per calcolare l'efficienza finale del modello.

- **Analisi del training set**
  I valori di probabilità relativi alla distribuzione delle lettere all'interno di *gt_tweets.csv* sono divisi in tre strutture dati:

  1. vettore di probabilità iniziali
     questo vettore, chiamato *pigreco*, contiene per ogni singola lettera la sua probabilità di rappresentare uno stato iniziale, ovvero di essere la prima lettera in una parola. È stato calcolando scorrendo tutte le parole del training set, e contando le occorrenze di ogni loro prima lettera.
  2. vettore di probabilità finali
     analogamente all' array illustrato in precedenza, questo tiene conto della probabilità di ogni lettera di rappresentare l'ultimo carattere di una parola.
  3. matrice delle probabilità di transizione
     la matrice di dimensione 26x26 *transiction_p* indica per ogni lettera, la sua probabilità di essere preceduta da una altra lettera dell'alfabeto. È stata calcolata analizzando a due a due tutte le coppie di lettere vicine nel testing set.

- **Inserimento e analisi degli errori**
  Utilizzando ancora una funzione randomica, è stato modificato il testing set introducendo il 10% di errori nei sui caratteri nel seguente modo:

$$\{\forall\, l \in\ A,\ \exists c \in mistakes(l),\ \exists r \in [0;1] \mid\ r < 0.1 \rightarrow sub(l,c)\}$$

  Dove:

  - A è l'insieme delle lettere presenti nei tweets;
  - *mistakes(l)* è l'insieme di tutti i caratteri che possono essere inseriti per errore al posto del carattere l;
  - r è un numero *random* compreso tra 0 e 1;
  - *sub* è una funzione che sostituisce il carattere l con c.

Si è poi proceduto a creare la matrice *obs_p*, che scorrendo in parallelo il testing set originale e quello contenente gli errori, definisce la probabilità di ogni lettera di rimanere tale dopo il processo di perturbazione oppure di essere sostituita con un altro carattere.

- **Creazione del modello**
  Utilizzando *Pomegranate* è stato possibile creare un modello di Markov appartenente al primo ordine.
  Per la definizione degli *stati* (le lettere dell'alfabeto), delle *osservazioni* (i possibili caratteri di errori associati a una lettera del tweet) e delle *transizioni* (i vari vettori e matrici illustrati in precedenza) sono stati utilizzati due cicli *for* per permettere al programma di creare il modello in modo completamente automatizzato.
  Un esempio illustrativo di *HMM* generato contenente solo 3 lettere è mostrato nella seguente figura:

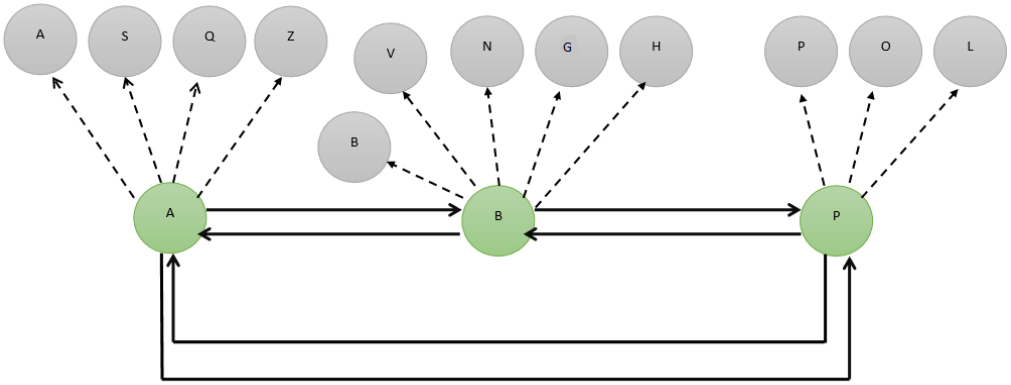

Figure 1: Esempio di un modello HMM con 3 caratteri - le probabilità di transizione sono state omesse per semplificare la lettura

- **Fase finale**
  Una volta creato il modello, gli si è dato in input il testing set per testarne le effettive capacità. Per ogni riga del documento si è quindi usato l'algoritmo di Viterbi su ciascuna parola, di modo che calcolasse la più alta probabilità di lettere che la formassero e che quindi correggesse eventuali errori. Al termine dell'algoritmo è stato quindi creato un nuovo file *output_tweets.csv* che contiene il testing set corretto.

Finita questa prima fase di ricerca, è iniziata quella più sperimentale, in cui si è implementata una semplice interfaccia grafica che permettesse di correggere attraverso il nostro algoritmo del testo in input immesso da un utente tipo.

1 Sviluppo dell'applicazione

- **Implementazione interfaccia**

  La GUI generata è composta dai seguenti elementi:

  1. **TextBox**: All'interno di questa *TextBox* viene inserito il testo dell'utente, cioè l'input al nostro programma;

  2. **PushButton**: Una volta inserito il testo nella *TextBox 1*, premendo *improve*, il programma inizierà ad utilizzare l'algoritmo di *Viterbi* sulla frase scritta dall'utente;

  3. **TextBox**: In questa *TextBox* viene mostrato l'output del nostro programma, ovvero la frase dell'utente corretta utilizzando il modello *HMM* del primo ordine e l'algoritmo di *Viterbi*;

Figure 2: GUI del progetto

  4. **Console**: Nella *Console* vengono mostrati due *KPI*:
     - Errors Before: dove viene indicata la percentuale di errori nei tweets prima del lancio del programma;

– Errors After: dove viene indicata la percentuale di errori presente nei tweets dopo aver lanciato l'algoritmo di *Viterbi* sul nostro modello *HMM*.

Il valore Q, definito come $Q = ErrorsBefore - ErrorsAfter$, indica la percentuale di errori che il nostro modello *HMM* è riuscito a correggere.
Oltre a questi *KPI*, la *Console* mostra anche messaggi di conferma per il salvataggio in locale dei file contenenti i tweets scaricati dal programma in seguito all'aver premuto i corrispondenti tasti;

5. **PushButton**: Il tasto *Input* permette di scaricare il *CSV* dei tweets in input al nostro programma, ovvero il file *lp_tweets.csv*;

6. **PushButton**: Il tasto *Output* permette di scaricare l'output del nostro programma, ovvero tutti i tweets corretti tramite l'algoritmo di *Viterbi*, in formato *CSV*.

- **Correzione input utente esterno**

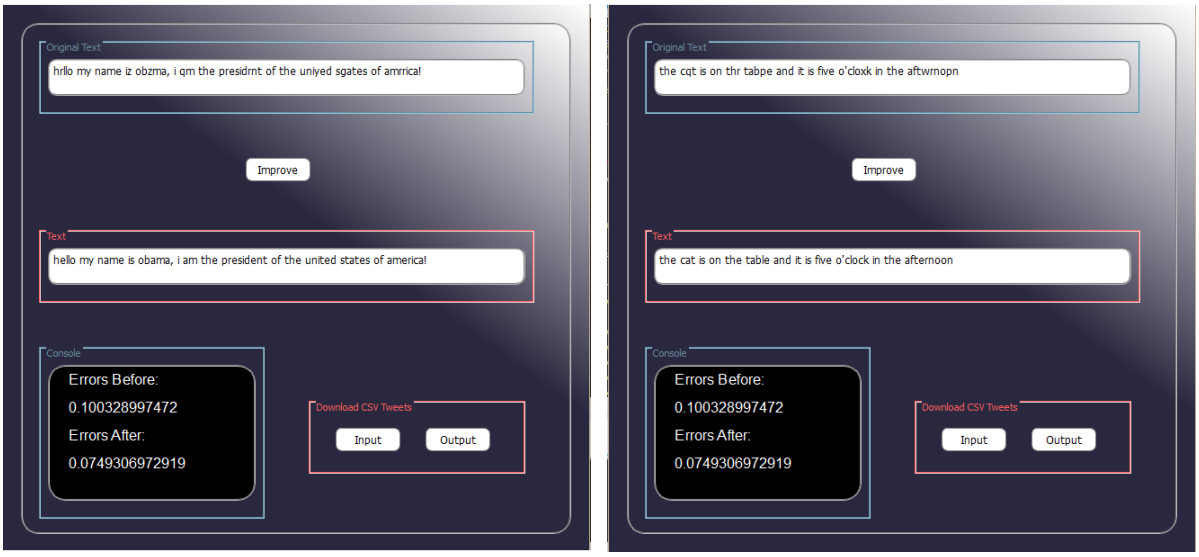

Figure 3: due output esempio che mostrano l'efficacia dell'applicazione

Per correggere un input generico immesso nella TextBox si è usato il già illustrato modello, strutturato con le precedenti distribuzioni probabilistiche, e si è fatta inferenza sfruttando il noto algoritmo di Viterbi.

Una notevole differenza è costituita però dal fatto che ora il testo accetta in input anche segni di punteggiatura e numerici, non più solo alfabetici. Questo è stato reso possibile attraverso un algoritmo ricorsivo creato *ad hoc*, che presa la stringa in input la spezza in due parti, la prima contenente tutti i caratteri precedenti al carattere *non-alfabetico* trovato, e la seconda contenente tutto il resto. Sulla prima parte viene quindi eseguito l'algoritmo di *Viterbi*, su quella restante viene ricorsivamente richiamata la funzione. Infine viene ricostruita la stringa output finale giustapponendo le varie sezioni in cui si divide l'input e i segni *non-alfabetici* rilevati ma non modificati.

Conclusioni

- **Obiettivi Raggiunti e Analisi**
  Analizzando le differenze tra i file *lp_tweets.csv* e *output_tweets.csv* si è potuta calcolare la capacità correttiva del programma. Questa si attesta in un valore che si aggira tra il 25% e il 26%; ovvero partendo dal file perturbato originale contenente il 10% di errori, il file finale ne contiene circa solo il 7.4%. Questo risultato può sicuramente variare ed essere migliorato cambiando alcune scelte implementative.

  – Un primo fattore che incide nel risultato finale è certamente la quantità di tweets usati come base iniziale per il calcolo delle probabilità. Al crescere dei profili da cui si vengono scaricati i dati, aumenta infatti la fonte di informazioni su cui fare analisi, e quindi le distribuzioni di probabilità ottenute risulteranno più precise e raffinate.
  D'altro canto, diminuendo il numero di tweets si assiste a due conseguenze opposte. Ipotizzando il caso estremo in cui sfruttassimo solo i tweets di un unico utente, da un lato la capacità predittiva dell'applicazione che si interfaccia con un utente calerà notevolmente, in quanto sfrutta valori di transizione più "grezzi". Dall'altro il valore di predizione fornito dal testing set sarà probabilmente cresciuto: questo perchè il singolo utente mantiene sempre un personale modo di scrivere e utilizza un personale vocabolario limitato di parole, quindi i dati del modello risulteranno "cuciti apposta per lui" e la correzione sarà più efficace.

  – Un secondo fattore corrisponde con la costruzione del modello: estendere questo ad un ordine superiore al primo ne migliora le capacità predittive. In questo modo infatti la probabilità di una

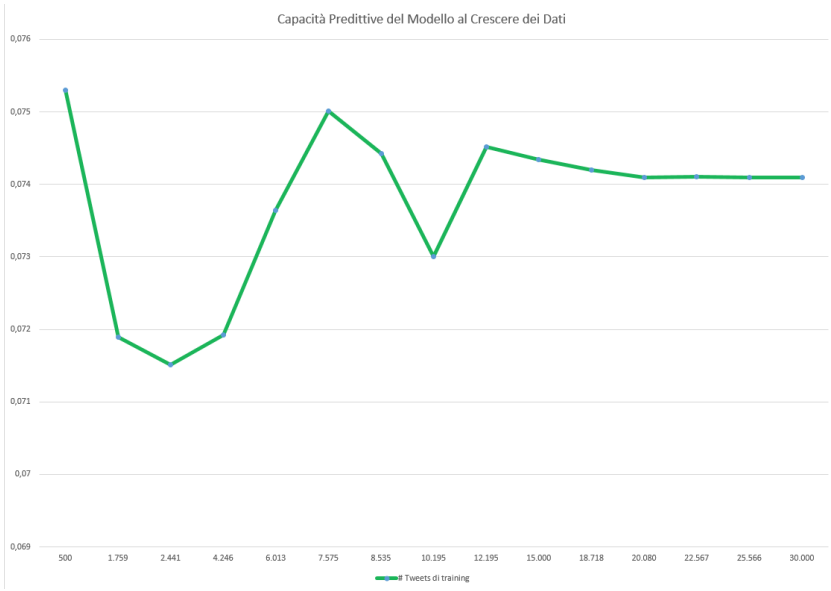

Figure 4: Capacità Predittiva al crescere del training set:
L'immagine mostra chiaramente come, all'aumentare del numero dei tweets, la correzzione del modello generato diventi asintotico alla retta passante per il punto y = 0.074

certa lettera di trovarsi in un determinato punto di una parola non dipenderà più solo dalla lettera precedente, ma anche da quella prima ancora o da altre n prima, a seconda dell'ordine *n-esimo* utilizzato.

– Infine un miglioramento leggero può essere portato dalla scelta delle fonti tweets. La nostra decisione di utilizzare determinati utenti potrebbe aver portato a performances diverse da quelle che si avrebbero avuto scegliendo i profili attraverso altri criteri.

Manuale Utente

Per aprire il progetto è necessario avviare il software *Eclipse* in modalità amministratore e avviare il file *main.py* python. Il programma procederà a ottenere i tweets, analizzarli e a creare le statistiche e il modello. Finito questo verrà mostrata l'interfaccia che permette di correggere del testo generico. L'utente potrà eventualmente cambiare fonte dei tweets o perfino cambiare lingua di correzione scegliendo differenti account da cui attingere i dati.