

ALMA MATER STUDIORUM · UNIVERSITÀ DI  
BOLOGNA

---

SCUOLA DI SCIENZE  
Corso di Laurea in Informatica

**Monitoraggio degli eventi di touch  
su dispositivi mobili utilizzando  
sensori inerziali**

**Relatore:**  
Dott.  
Luca Bedogni

**Presentata da:**  
Andrea Alcaras

**Sessione III**  
**Anno Accademico 2017/2018**



# Introduzione

Al giorno d'oggi i dispositivi mobili sono un elemento sempre piú presente nella vita quotidiana di tutti gli individui. Tali dispositivi hanno subito un'evoluzione hardware e software tale da renderli degli " *smartphone*" e da permettere il loro utilizzo per gli scopi piú svariati. Infatti, i moderni cellulari, oltre a permettere di comunicare tramite chiamate e messaggi, sono dotati di applicazioni che permettono agli utenti di svolgere svariate operazioni inerenti all'ambito quotidiano. Eppure, queste applicazioni, oltre a semplificare molte azioni quotidiane, mettono a rischio i dati sensibili degli utenti, come *passwords* e *accounts*. Ad esempio, questi dati sensibili possono essere messi a rischio da attacchi di *spoofing*, in cui chi effettua l'attacco, inserendosi all'interno di una rete, simula l'identitá di un *host* o di un mittente in modo da poter intercettare informazioni sensibili. Nel contesto di questo progetto, tuttavia, verrá considerato un diverso tipo di attacco, il quale avviene prima della trasmissione dei dati da parte del dispositivo mobile. L'idea che si cela dietro questo tipo di attacchi prende origine dalla considerazione che, al giorno d'oggi, tutti i dispositivi mobili sono dotati di sensori quali accelerometro, magnetometro, giroscopio e altri, i cui valori possono essere letti senza la necessitá di richiedere permessi. Inoltre, i valori registrati da tali sensori hanno lo scopo di determinare l'orientamento e i movimenti del dispositivo nello spazio, in modo tale da permettere a chiunque abbia accesso a tali dati di determinare gli spostamenti, anche quelli piú impercettibili, del dispositivo nel tempo. In particolare, verrá preso in considerazione il caso in cui il dispositivo viene utilizzato con una sola mano: tale operazione implicherá un

movimento piú ampio da parte dell'utente, poiché per effettuare il *touch* sullo schermo sarà costretto a piegare il dispositivo. Quindi, si potrebbe postulare l'esistenza di un'applicazione in grado di leggere e interpretare i valori di tali sensori. Tale applicazione potrebbe anche essere in grado di interpretare quali caratteri sono stati selezionati dall'utente, grazie ad un'analisi di tali dati da parte di un modello di *machine-learning* precedentemente addestrato a tale scopo. Le informazioni ottenute tramite lo studio di tali dati potrebbero, inoltre, permettere ad un'applicazione malevola di riconoscere non solo le abitudini di scrittura relative all'utente, ma anche altre abitudini che fanno parte della sua vita quotidiana: ad esempio, la velocità con cui vengono digitati i tasti, la frequenza dei *touch events* e le parole piú frequentemente usate dall'utente stesso. Ovviamente, piú sarà ampia la porzione di schermo corrispondente all'evento *touch*, piú risulterà semplice interpretare tali dati. Nel contesto di questo elaborato, verranno considerati, in particolare, i casi in cui lo schermo venga diviso in: 8, 16, 32, e KBD sezioni. Inoltre, ci si soffermerà principalmente sullo studio del caso KBD, corrispondente alla divisione dello schermo in tante sezioni quante sono quelle presenti in una tastiera virtuale di un dispositivo mobile. Infatti, l'analisi di questo caso in particolare permetterà di mostrare la reale possibilità di riconoscere i tasti digitati dall'utente sul dispositivo, al fine di interpretare dati privati e non digitati dall'utente in questione. Un'ulteriore distinzione importante va effettuata anche tra i due casi di studio trattati in questo elaborato, ovvero tra il caso in cui si conosca a priori l'orientamento e la posizione relativa alla digitazione dei caratteri sul dispositivo da parte dell'utente e il caso piú complesso in cui, invece, sarà necessario effettuare un ricalcolo di tali dati, al fine di ricavare le informazioni necessarie al riconoscimento dei *touch events* dai dati stessi.

Infine, è possibile affermare che tale progetto è stato sviluppato in tre fasi. La prima fase, relativa alla parte *mobile*, si occupa di registrare i valori dei sensori precedentemente citati e le relative porzioni di schermo selezionate. La seconda fase tratterà dell'analisi delle *features* relative ai dati da parte di

un modello di *machine learning*, il quale riconoscerà successivamente le varie porzioni di schermo selezionate. Infine, la terza fase si occuperà di dimostrare quanto sia effettivamente possibile riconoscere i caratteri e quindi le parole digitate dall'utente, con una percentuale di successo pari all'80% per parole di pochi caratteri e con una percentuale di successo maggiore all'80% per parole più lunghe.



# Indice

<b>Introduzione</b>	<b>iii</b>
<b>1 Stato dell'arte</b>	<b>1</b>
1.1 Sensori di movimento . . . . .	2
1.2 Context-aware-computing . . . . .	5
1.3 Privacy e dispositivi mobili . . . . .	7
<b>2 Applicazione</b>	<b>11</b>
2.1 Activity principale . . . . .	12
2.2 Activity SensorEventListener . . . . .	13
2.2.1 Azioni OnCreate . . . . .	13
2.2.2 Salvataggio dati sensori . . . . .	16
<b>3 Modello</b>	<b>21</b>
3.1 Touch events e calcolo delle features . . . . .	22
3.1.1 Isolare i touch events . . . . .	22
3.1.2 Calcolo delle features . . . . .	22
3.2 Modello . . . . .	25
3.2.1 Background . . . . .	25
3.2.2 Implementazione del modello e casi considerati . . . . .	26
<b>4 Risultati</b>	<b>29</b>
4.1 Risultati relativi alle predizioni effettuate dalle <i>features</i> dell'accelerometro . . . . .	30

4.2	Risultati relativi alle predizioni effettuate dalle <i>features</i> di accelerometro e magnetometro . . . . .	31
4.3	Risultati relativi al ricalcolo dei dati grezzi . . . . .	32
4.4	Risultati relativi alle predizioni delle <i>features</i> ottenute dal ricalcolo di accelerometro e magnetometro . . . . .	34
4.5	Risultati per i casi: 8, 16, 32, KBD . . . . .	35
4.6	Riconoscimento delle parole digitate . . . . .	37
4.7	Risultati relativi al dizionario . . . . .	41
<b>5</b>	<b>Conclusioni</b>	<b>49</b>
	<b>Bibliografia</b>	<b>51</b>



# Elenco delle figure

2.1	Architettura applicazione . . . . .	11
2.2	Main activity . . . . .	14
2.3	Sensor Event Listener . . . . .	15
2.4	Dati grezzi Accelerometro per i tre assi nel tempo, 1 click per sezione nel caso di 8 sezioni. . . . .	19
3.1	Eventi touch Accelerometro, 1 click per sezione nel caso di 8 sezioni. . . . .	24
3.2	Variazione media(Avg) nel tempo del caso mostrato in Figura 3.1. . . . .	24
3.3	Categorie dei modelli di <i>Machine-learning</i> . . . . .	25
4.1	Grafico relativo all'accuratezza delle predizioni per 8 sezioni, nel caso in cui vengano considerate le <i>features</i> dell'accelerometro inerenti ad una singola registrazione. . . . .	30
4.2	Grafico relativo all'accuratezza delle predizioni per 8 sezioni, a partire dai dati relativi ad accelerometro e magnetometro all'interno di una singola registrazione. . . . .	31
4.3	Grafico sull'accuratezza delle predizioni per 8 sezioni di schermo inerenti ai dati grezzi dell'accelerometro nell'arco di 10 registrazioni. . . . .	32
4.4	Grafico sull'accuratezza delle predizioni per 8 sezioni inerenti ai dati ricalcolati dall'accelerometro nell'arco di 10 registrazioni. . . . .	33

4.5	Grafico relativo all'accuratezza delle predizioni per 8 sezioni inerenti alle <i>features</i> di accelerometro e magnetometro ed effettuare a partire dai dati ricalcolati nell'arco di 10 registrazioni.	34
4.6	Grafico a barre relativo ai migliori risultati ottenuti dai dati di una registrazione, nel caso in cui si consideri solo l'accelerometro <i>in rosso</i> e il caso in cui venga aggiunto anche il magnetometro <i>in verde</i> .	35
4.7	Grafico a barre relativo ai migliori risultati ottenuti e inerenti ai dati di 10 registrazioni, nel caso in cui venga considerato solo l'accelerometro <i>in rosso</i> e nel caso in cui si consideri anche il magnetometro <i>in verde</i> .	37
4.8	Esempio in cui $C_i = a$ e $A(C_i) = q, w, s, z$ .	38
4.9	Albero delle possibilità per la parola <i>App</i> .	39
4.10	Frequenza delle parole in relazione alla lunghezza del dizionario considerato.	42
4.11	<i>Distanza di Levenshtein</i> Media.	43
4.12	Media <i>rank</i> della parola cercata.	43
4.13	Accuratezza del riconoscimento.	44
4.14	Distanza di Levenshtein media	46

# Capitolo 1

## Stato dell'arte

Al giorno d'oggi tutti i dispositivi mobili sono dotati di sensori, lo scopo di tali sensori é quello di migliorare l'esperienza dell'utente basandosi sulle rilevazioni fornite dal dispositivo relativamente all'ambiente e al modo in cui l'utente interagisce con esso e con il dispositivo. I sensori di cui sono dotati i dispositivi mobili odierni sono suddivisibili in tre macro categorie:

- **Sensori di movimento**, i quali si occupano di rilevare i cambiamenti dei valori relativi all'accelerazione e alla gravitá lungo i tre assi;
- **Sensori di posizione**, i quali si occupano di rilevare dati relativi alla posizione fisica del dispositivo;
- **Sensori di ambiente**, cosí detti perché rilevano i valori relativi all'ambiente circostante, come quelli relativi alla temperatura, alla pressione e umiditá.

Grazie a questi sensori é quindi possibile percepire i cambiamenti dell'ambiente che circonda i vari dispositivi mobili di cui oggi disponiamo in sempre maggior misura. In particolare i sensori di movimento hanno la funzione di rilevare i movimenti nello spazio del dispositivo, rendendo possibile la rilevazione di vari dati sensibili dell'utente come verrá mostrato nelle prossime sezioni.

## 1.1 Sensori di movimento

Nel contesto di questo elaborato, particolare importanza verrà attribuita ai sensori di movimento, in quanto sono i sensori utilizzati all'interno del progetto al fine di identificare gli eventi touch avvenuti sullo schermo del dispositivo.

La maggior parte dei dispositivi moderni é ormai dotata di tali sensori e la loro presenza rende possibile il riconoscimento degli spostamenti nello spazio del dispositivo e conseguentemente anche dell'utente. Alcuni sensori di movimento particolarmente sfruttati dai moderni dispositivi mobili sono:

- Accelerometro, il quale permette di calcolare e identificare la forza di accelerazione a cui viene sottoposto il dispositivo;
- Magnetometro, che grazie alla rilevazione del campo magnetico é in grado di stabilire l'orientamento del dispositivo;
- Giroscopio, il quale distingue i movimenti e le rotazioni avvenute sul dispositivo lungo i tre assi nello spazio.

Grazie all'analisi dei dati immagazzinati da questi sensori é possibile riconoscere quali siano le attività svolte dall'utente. Ad esempio, in [1] viene mostrato come, partendo dai dati ottenuti dalle registrazioni di accelerometri posizionati sui soggetti studiati, sia possibile riconoscere attività svolte quotidianamente dall'individuo, come ad esempio camminare o correre. Lo studio contenuto nell'articolo citato evidenzia, inoltre come, sfruttando due soli sensori di accelerazione posti su polso e caviglia, questi risultati siano soggetti ad un leggero decremento. Inoltre, grazie all'analisi dei dati rilevati dai sensori di movimento, é anche possibile individuare quale sia il mezzo utilizzato dall'utente in questione per effettuare i suoi spostamenti. La possibilità di individuare persino il mezzo di spostamento viene mostrata negli articoli [2, 18]. In particolare, nell'articolo [18] viene definito il metodo tramite cui può avvenire tale attacco. Partendo da un dataset generato dalle registrazioni dei sensori effettuate su 13 utenti e reso pubblico per ricerche future,

vengono effettuati i test necessari al riconoscimento del mezzo di trasporto utilizzato dall'utente. Questi test evidenziano anche che i sensori piú importanti per il riconoscimento del mezzo di trasporto utilizzato dall'utente sono l'accelerometro e il giroscopio. Infine, l'articolo mostra come un modello di *machine-learning* addestrato sia in grado, a partire da tali dati, di riconoscere con alta precisione il mezzo utilizzato dall'utente. All'interno dell'articolo [2] viene ulteriormente confermata la possibilità di carpire tale informazione dall'utente, partendo dai dati ottenuti da accelerometro e giroscopio. Inoltre, verrà utilizzato a tale scopo anche un modello di *machine-learning* adeguatamente addestrato al fine di analizzare i dati registrati dall'accelerometro e dal giroscopio del dispositivo considerato, senza la necessità di sfruttare dati provenienti dal GPS. L'articolo mostra anche come sia possibile ottenere gli stessi risultati relativi alla predizione del mezzo di trasporto al pari di un minore consumo di energia del dispositivo, ottenuto tramite l'utilizzo dei dati relativi ai sensori citati e non ai dati ottenuti tramite il GPS. Invece, in [7] viene mostrato come sia anche possibile distinguere le abitudini del singolo individuo relative al suo modo di camminare. Tutto ciò é reso possibile dall'analisi dei dati relativi alla falcata, alla velocità e alla frequenza del passo dell'utente considerato. Questi dati sono fruibili grazie all'uso di un semplice accelerometro e a partire da essi risulterà possibile addestrare un modello di *machine-learning* al riconoscimento dell'individuo con ottime percentuali di successo.

Quando ci si avvicina allo studio dei dispositivi mobili dotati di sensori di movimento non va inoltre sottovalutata la presenza di "*wearable devices*", come gli *smartwatch*, i quali possono anche essere sfruttati allo scopo di carpire dati sensibili dall'utente. Infatti, grazie ai sensori di cui essi sono dotati, anche gli *smartwatch* possono tener traccia dei movimenti del polso dell'utente. Nel caso in cui un'applicazione malevola avesse accesso a questi dati, sarebbe possibile ottenere una discreta accuratezza nel riconoscimento dei caratteri scritti a mano su un foglio dall'utente. La reale possibilità di sfruttamento di tali informazioni da parte di un'applicazione malevola che si affida, per

l'analisi dei dati in questione, ad un modello di *machine-learning*, viene mostrata in [16]. Nell'articolo [9] viene evidenziato come, sfruttando lo stesso principio di [16], sia possibile sfruttare i dati relativi allo spostamento del polso di cui lo *smartwatch* tiene traccia, allo scopo di interpretare i tasti premuti dall'utente su una tastiera fisica QWERTY. Tutto ciò é reso possibile grazie allo studio dei dati ottenuti dai sensori inerziali dello *smartwatch* e relativi alla posizione fisica dei tasti digitati e agli spostamenti necessari alla digitazione di tasti consecutivi tra loro. Sulla base dello stesso principio presentato nell'articolo [9], in [13] viene evidenziata la possibilità di riconoscere i tasti premuti su una tastiera generica, sia appartenente ad uno smartphone sia appartenente, ad esempio, ad un ATM, al fine di individuare la tastiera di riferimento. Questa possibilità può realmente verificarsi grazie allo studio dei pattern di movimento del dispositivo e al riconoscimento di determinati pattern di movimento, a partire dai dati relativi ai sensori inerziali presenti nello *smartwatch*. All'interno dell'articolo citato vengono anche presentati alcuni semplici accorgimenti utili ad evitare queste situazioni, come ad esempio la disattivazione dei sensori inerziali in prossimità di un ATM, al fine di escludere totalmente la possibilità di ricezione dei dati da parte di un'applicazione malevola.

Abbiamo quindi visto come, sfruttando i sensori di movimento dello *smartwatch* indossato dall'utente considerato, sia possibile riconoscere gli eventi touch di uno *smartphone*. Nell'articolo [10] viene proprio evidenziato come tali dati possano essere sfruttati da un'applicazione malevola se non adeguatamente salvaguardati. Infatti, grazie al rilevamento dei movimenti del polso risulta possibile riconoscere i tasti digitati sul tastierino numerico incluso nello *smartphone*.

Dopo aver analizzato vari casi di sfruttamento di dati rilevati dai sensori di un *wearable device*, in particolare di uno *smartwatch*, ci si può adesso soffermare sullo studio dei sensori presenti solo sullo smartphone. Tali sensori possono essere chiaramente utilizzati in modo simile ai casi precedentemente descritti, al fine di ricavare informazioni sensibili relative all'utente che

utilizza lo *smartphone* in analisi. Ad esempio, negli articoli [5, 14] viene dimostrata la possibilità di comunicazione tra due dispositivi, senza che questi siano fisicamente a contatto. Questa comunicazione indiretta risulta possibile grazie alle vibrazioni emesse da uno dei due dispositivi e alla ricezione di queste vibrazioni da parte dell'accelerometro dell'altro dispositivo. Infatti, due dispositivi posizionati su una superficie in comune, ad esempio una scrivania, possono comunicare grazie alla vibrazione emessa dallo smartphone intento a comunicare, la quale viene a sua volta recepita dall'accelerometro del dispositivo in attesa dell'informazione. Nel lavoro presentato in [15] viene mostrato, invece, come sia possibile modificare i valori registrati dall'accelerometro sfruttando delle specifiche tracce audio in grado di disturbare tali registrazioni di dati, al fine di rendere più difficile l'ipotetico riconoscimento di specifici pattern e dei tasti digitati sullo schermo dall'utente considerato. Più vicino al caso considerato in questo elaborato è, invece, il lavoro presentato nell'articolo [11], il quale evidenzia la possibilità di riconoscere intere sequenze di caratteri digitate su uno smartphone, partendo dai dati relativi all'accelerometro e ottenuti durante l'esecuzione di eventi touch sullo schermo del dispositivo, al fine di scoprire la password dell'utente.

Nel lavoro descritto all'interno di questo elaborato, oltre ai dati relativi all'accelerometro, verranno utilizzati anche i dati relativi al magnetometro, i quali permettono di ottenere una maggiore precisione nel riconoscimento degli eventi di touch sullo schermo. Inoltre, non verrà studiato il caso singolo della digitazione di una password, ma il caso della digitazione di una serie di caratteri atti a formare parole di senso compiuto e valutate a partire dall'utilizzo di un dizionario in lingua inglese.

## 1.2 Context-aware-computing

Il contesto comprende tutte le informazioni che possono essere rilevanti al fine di identificare lo stato di una data entità nel tempo. Alcuni esempi di dati relativi al contesto sono:

- Le attività svolte dall'utente considerato;
- L'identità dell'utente;
- I dispositivi nelle vicinanze;
- Il tempo.

I dati relativi al contesto possono anche essere rappresentati da semplici rilevazioni relative all'ambiente circostante e ottenute grazie alle componenti hardware del dispositivo. Esempi di tali rilevazioni possono essere:

- La luce a cui viene esposto il dispositivo;
- La posizione sia GPS, sia lungo i tre assi dello spazio;
- La pressione a cui viene sottoposto il dispositivo.

Alla base di questo principio di interazione tra l'uomo, l'ambiente circostante e il dispositivo vi é il concetto di *context-awareness*, il quale ha l'obiettivo di fornire un'esperienza personalizzata all'utente, utilizzando sia i dati relativi al contesto sia quelli relativi alle modalità tramite cui l'individuo interagisce con il proprio dispositivo. Quindi, il *context-aware-computing* si basa sull'uso di dispositivi hardware e tecniche software utili a collezionare i dati necessari al dispositivo, permettendogli di reagire nel modo piú adatto. Alcuni semplici esempi di *context-aware-computing* sono:

- Il riconoscimento del passaggio dalla modalità *landscape* a quella *portrait* e viceversa;
- La riduzione o l'aumento della luminosità dello schermo in relazione all'esposizione luminosa a cui é sottoposto il dispositivo;
- La disattivazione dello schermo nel momento in cui, durante una chiamata, il dispositivo viene avvicinato all'orecchio. Questa disattivazione é utile al fine di evitare il possibile tocco involontario dei tasti dello schermo tattile.



Lo sviluppo del concetto di *context-aware-computing* ha permesso di ottenere un miglioramento nell'approccio rispetto a diverse problematiche, come quella che ha dato vita ad un altro concetto molto importante nell'ambito delle interazioni tra l'uomo e il dispositivo: il concetto di *E-health* [6].

L'*E-health*, complesso di risorse e tecnologie informatiche applicate al tema della salute e della sanità, dispone di una serie di dispositivi dotati di particolari sensori appositamente utilizzati, ad esempio, per le rilevazioni mediche, grazie alle quali é possibile monitorare il paziente e riconoscere, a partire dai dati ottenuti, alcune delle sue abitudini.

Inoltre, é possibile affermare che la capacità dei moderni dispositivi mobili di immagazzinare dati relativi alla vita privata dell'utente implica automaticamente la possibilità che questi dati, rilevati grazie all'uso dei sensori, vengano utilizzati da applicazioni malevole, le quali possono carpire tramite questi dati informazioni sensibili relative all'utente a sua insaputa. Questo é il caso mostrato nell'articolo [19], in cui viene inizialmente presentata la possibilità di interpretare azioni future dell'utente sotto attacco da parte di un'applicazione malevola sulla base dei dati relativi al contesto in cui un'utente compie determinate azioni. Successivamente viene però evidenziata una soluzione a tale problema che sfrutta l'utilizzo di un *framework*. Quest'ultimo, diversamente dall'approccio tradizionale in cui si é soliti nascondere una parte dei dati relativi al contesto, é in grado di aggiungere del "rumore", ovvero dati superflui ai dati intercettati da colui che esegue l'attacco. Questo processo di aggiunta di dati superflui riesce a rendere più difficile l'interpretazione del contesto, a partire dai dati alterati che vengono rilevati da colui che esegue l'attacco.

### 1.3 Privacy e dispositivi mobili

Sulla base delle osservazioni effettuate nei paragrafi precedenti, risulta quindi evidente come oggi il costante utilizzo di smartphone da parte della maggior parte degli individui evidenzia un problema di privacy relativa al

trattamento dei dati ottenibili dai dispositivi stessi. Una soluzione a questo problema viene offerta dalla possibilità di poter rendere anonimi i propri dati. Tuttavia, è impossibile pensare di garantire ad un qualsiasi utente il completo anonimato: molti studi hanno infatti dimostrato che anche da dati anonimi è possibile ricavare informazioni specifiche relative dell'utente in questione. Ad esempio, nell'articolo [3] viene mostrata la possibilità di riconoscere la posizione dell'abitazione dell'utente e il relativo luogo di lavoro, a partire da una serie di dati GPS ottenuti dal dispositivo. Viene però anche evidenziata la possibilità di rendere anonimi e più difficili da interpretare tali dati al prezzo di una minore precisione della localizzazione del dispositivo. Lo studio e la classificazione dei dati considerati fin'ora viene effettuato sfruttando modelli di *machine-learning* adeguatamente addestrati al riconoscimento di determinate abitudini e comportamenti propri dell'utente. Un esempio di utilizzo di tali modelli è fornito dal *Anticipatory mobile computing*, il cui funzionamento è trattato, in particolare, nell'articolo [12]. Nell'articolo citato viene mostrato come dei modelli di *machine-learning*, a partire da azioni avvenute nel passato, rappresentate in forma di dato e classificate, siano in grado di predire con una certa accuratezza azioni future. Un esempio tipico di *Anticipatory mobile computing*, che potrebbe entrare a far parte della vita quotidiana dell'utente, è rappresentato dalla predizione dei futuri spostamenti dell'utente stesso, sulla base dell'analisi delle sue azioni e dei suoi spostamenti passati. Quindi, basandosi sul principio dell'*Anticipatory mobile computing* e utilizzando una serie di dati relativi alla posizione dell'utente nel tempo è anche possibile riconoscere determinate azioni o spostamenti effettuati dall'utente. Similmente, in [4, 8] viene studiato come, a partire dai dati pubblicamente disponibili sui *social network*, tra cui informazioni relative all'utente specifico e informazioni provenienti dalle relazioni sviluppate all'interno del *social network* con altri utenti, sia possibile riconoscere la provenienza dell'utente preso in considerazione. L'articolo [8] sfrutta, in particolar modo, i dati relativi alle informazioni rese pubbliche su *Twitter* al fine di individuare il luogo di provenienza dell'utente in analisi. Tuttavia, tali informazioni, come descritto

nell'articolo, raramente contengono informazioni relative alla localizzazione geografica. Quindi, l'idea alla base dello studio presentato nell'articolo [8] é quella di sfruttare le informazioni relative al contenuto dei *tweets* e alle abitudini dell'utente considerato ottenibili dagli stessi *tweets*. Un ulteriore esempio di acquisizione di dati sensibili tramite dispositivi mobili viene presentato in [17], nel quale viene mostrato come, ottenendo delle tracce audio relative alla scrittura su foglio eseguita da un individuo, sia possibile risalire alle parole scritte dall'utente stesso. Questa operazione comporta un problema di violazione della privacy non indifferente, in quanto il caso piú semplice da considerare é quello in cui un individuo compili dei documenti contenenti informazioni private all'interno un luogo pubblico. Al fine di ottenere tali dati un utente con intenti malevoli potrebbe registrare la traccia audio di tale scrittura su foglio per poi analizzarla successivamente, ottenendo cosí i dati personali dell'utente a cui appartiene la traccia audio considerata.



## Capitolo 2

# Applicazione

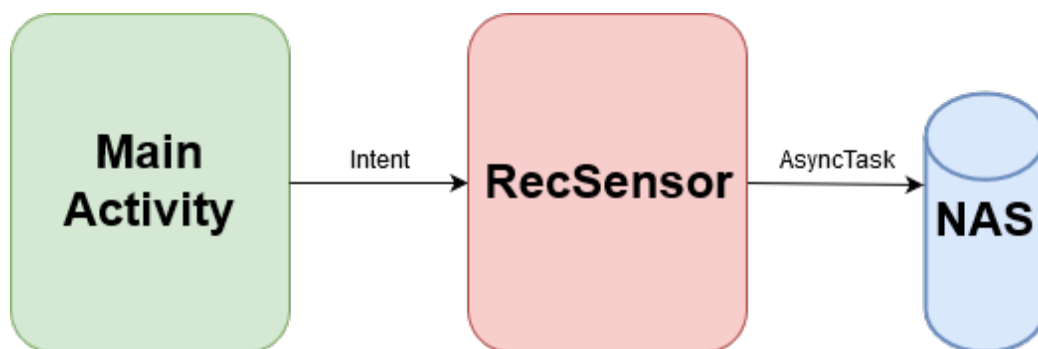


Figura 2.1: Architettura applicazione

Come già anticipato nell'*Introduzione* a questo elaborato, il progetto é articolato in piú fasi. La prima fase riguarda l'applicativo sviluppato per dispositivi Android. Tale applicazione puó esse utilizzata dai dispositivi mobili che supportano Android 6.0. É stata scelta questa particolare versione di Android in relazione al dispositivo utilizzato per effettuare le registrazioni dei sensori, ovvero un Asus Zenfone Max (Z010D). Inoltre, il dispositivo risulta essere privo del sensore relativo al giroscopio, motivo per cui i dati registrati da tale sensore non verranno trattati nella fase seguente e in quelle successive. Infine, é necessario dire che la funzione di questo applicativo é quella di raccogliere i dati relativi ai sensori, quali accelerometro e magneto-

metro, e alle sezioni di schermo interessate dall'evento touch. L'applicazione é composta da due *activities* principali. La prima si occupa di:

- Richiedere all'utente l'immissione di un nome usato per indicare il *file* che verrà creato;
- Selezionare in quante sezioni lo schermo verrà diviso.

Successivamente, questi dati vengono comunicati alla seconda *activity*, la quale si occupa di:

- Creare il *file* in cui tutti i valori dei sensori vengono registrati;
- Aggiornare questo *file* ogni qual volta un sensore registra un nuovo valore;
- Inviare, tramite una richiesta FTP asincrona, tale file a un *server NAS*, in modo da potervi facilmente accedere; e
- Eliminare il *file* se la richiesta andata a buon fine.

Dopo aver brevemente descritto il loro funzionamento, ci si può ora concentrare piú in particolare sull'analisi delle *activities* appena citate.

## 2.1 Activity principale

All'interno della prima *activity* é contenuta la classe *MainActivity* che, durante il primo avvio *onCreate*, richiede il permesso di poter accedere alla memoria del dispositivo. Questa operazione viene effettuata per comodità di accesso ai dati, poiché tramite la creazione di un *file CSV* all'interno del dispositivo, sarà successivamente possibile inviare il *file* stesso ad un *server* per analizzarne i dati. Al fine di evitare tale richiesta, un'applicazione malevola potrebbe richiedere un permesso di accesso ad internet inserendo all'interno del *file manifest* il permesso *android.permission.INTERNET* atto a comunicare i dati interessati.

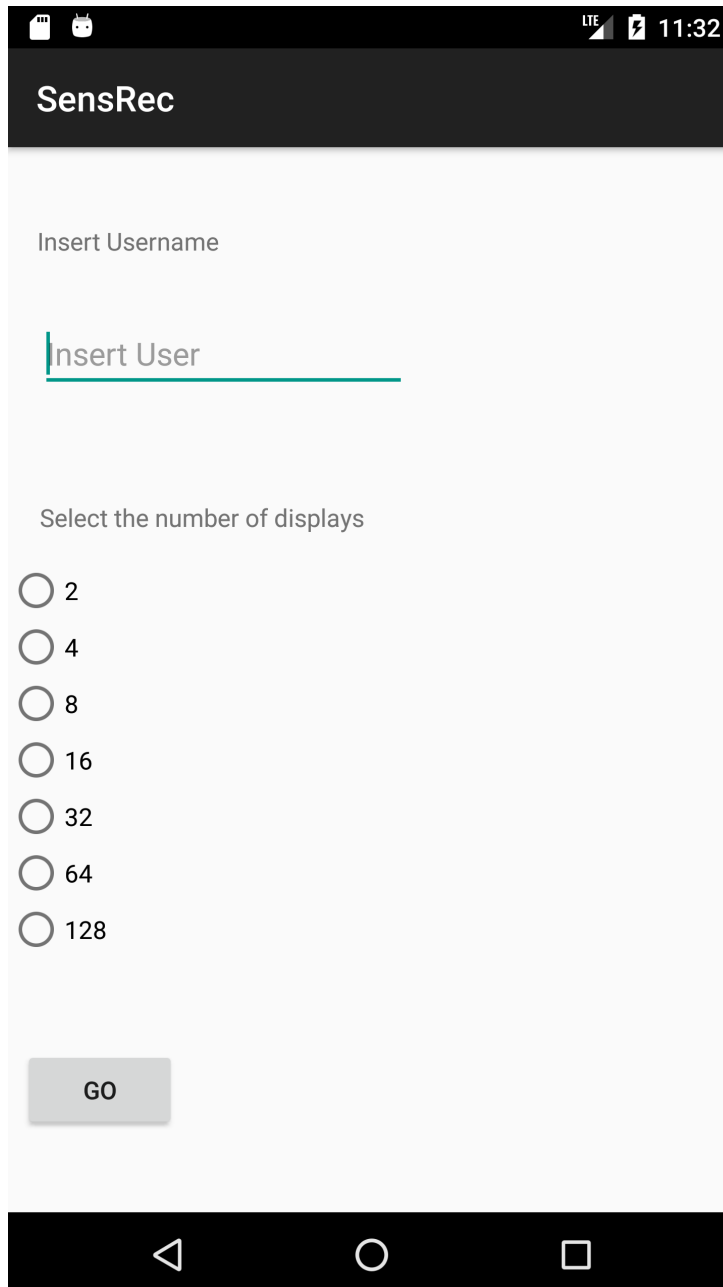
Successivamente, all'interno dell'*activity* viene caricato il *layout XML* relativo alla pagina iniziale, come evidenziato dalla Figura 2.2, la quale mostra un semplice *form* da compilare, al cui interno va inserito un nome utente, successivamente associato al *file* creato. Successivamente, sarà anche necessario selezionare dal *RadioGroup* sottostante una delle opzioni per poter procedere. Tali opzioni indicano il numero di sezioni in cui si vuole dividere il proprio schermo prima di avviare la registrazione dei sensori. Infine, cliccando sul *Button Go* verrà inizializzata la seconda *activity*.

## 2.2 Activity SensorEventListener

### 2.2.1 Azioni onCreate

La seconda *activity*, la cui classe *RecSensor* implementa il *SensorEventListener*, si occupa di registrare i valori ricevuti dai sensori in un *file* inizialmente contenuto all'interno della memoria del dispositivo e successivamente inviato al *server*. Il *SensorEventListener* è una classe fornita da *AndroidOS* che viene notificata dal *SensorManager* ogni qual volta i sensori interessati ricevono nuovi dati.

La classe, nel metodo *onCreate*, riceve un *intent* dall'*activity* principale: questo *intent* contiene il nome inserito dall'utente e il numero di sezioni in cui dividere lo schermo. Grazie a queste informazioni, l'*activity* principale è in grado di creare il *layout* generale dell'applicazione, aggiungendo le varie sezioni necessarie, come si può vedere dalla Figura 2.3, in cui viene mostrato il risultato nel caso in cui vengano selezionate 8 sezioni. A tal fine, viene caricato staticamente come *layout* generale il *file screens.XML* presente nella cartella delle *Resources*: questo *file* è formato da un semplice *LinearLayout* con allineamento verticale. È stato utilizzato il *LinearLayout* poiché permette di allineare tutti gli elementi presenti al suo interno in una determinata direzione e poiché rende possibile l'assegnazione di un *weight* ad ognuno degli elementi considerati. In questo modo è possibile stabilire che "importanza" assumono i vari elementi in termini di spazio all'interno del nostro *display*.



The screenshot displays the main activity of the SensRec application. At the top, a black header bar contains the app title "SensRec" in white. Below the header, the main content area is light gray. It features a text input field with the placeholder "Insert Username" and a green underline. Below this, there is a section titled "Select the number of displays" with seven radio button options: 2, 4, 8, 16, 32, 64, and 128. At the bottom of the form is a gray button labeled "GO". The Android navigation bar is visible at the very bottom.

Figura 2.2: Main activity



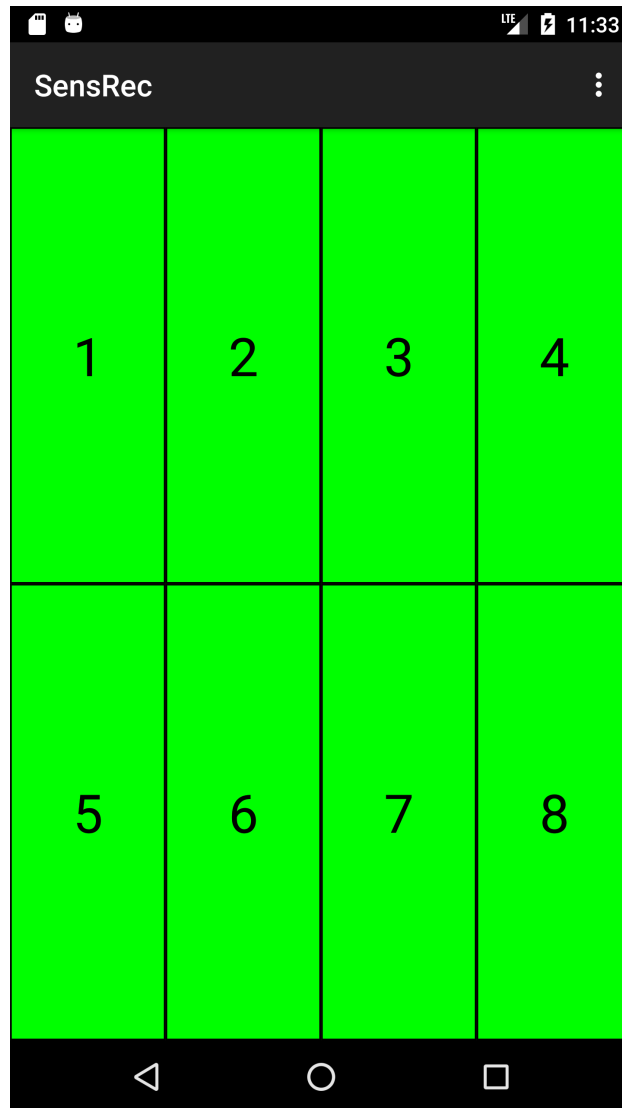


Figura 2.3: Sensor Event Listener

Dopo aver calcolato il numero di righe e colonne necessarie a dividere lo schermo in sezioni, vengono creati tanti *LinearLayout* quante sono le righe necessarie. Inoltre, ad ognuna di queste righe viene assegnato un orientamento orizzontale e un *weight* settato ad 1. All'interno di ogni riga vengono così creati tanti *LinearLayout* quante sono le colonne necessarie, in modo analogo al precedente *LinearLayout*. L'unica differenza consiste nel fatto che ad ogni *LinearLayout* viene assegnato un tag relativo al numero della sezione corrispondente e un metodo *OnTouchListener*. Quest'ultimo ha lo scopo di rilevare gli eventi *ACTION\_DOWN* e *ACTION\_UP* al fine di modificare il valore della variabile globale che tiene traccia della sezione di schermo cliccata, assegnandole un valore uguale al *tag* della sezione selezionata durante l'evento di touch *ACTION\_DOWN* e impostando un valore pari a -1 quando, invece, si verifica l'evento *ACTION\_UP*. Infine, all'interno di ogni sezione, viene inserita una *TextView* contenente il numero relativo alla sezione di schermo considerata.

Dopo aver generato il *layout*, l'applicazione crea un *file CSV* nella memoria del dispositivo usando come nome quello precedentemente inserito dall'utente e aggiungendo un *timestamp* in millisecondi relativo al momento in cui il *file* viene creato al fine di renderlo univoco. Infine, tramite il *SensorManager* vengono settate tre variabili che verranno usate per avviare le registrazioni e per gestire gli eventi relativi ai sensori che ci interesserà analizzare.

### 2.2.2 Salvataggio dati sensori

Nel contesto di questo applicativo, si terranno in considerazione solo alcuni sensori di cui sono provvisti i moderni dispositivi mobili, tra cui in particolare:

- L'accelerometro, il quale fa parte dei sensori di movimento;
- Il magnetometro, il quale appartiene alla categoria dei sensori di posizione.

Ai fini del progetto in questione, verrà tenuto in considerazione anche l'orientamento, ricalcolato a partire dai dati relativi all'accelerometro e al magnetometro nel modo seguente:

```
SensorManager.getRotationMatrix(mR, null, mLastAcc, mLastMag);  
SensorManager.getOrientation(mR, mOrientation);
```

In questo modo, tramite la funzione *getRotationMatrix* e a partire dai valori di accelerometro e magnetometro, viene calcolata la matrice di rotazione. Quest'ultima verrà poi successivamente utilizzata dal metodo *getOrientation* per calcolare l'orientamento del dispositivo. A questo punto, risulta interessante notare come, tramite la classe *SensorEventListener*, sia possibile rilevare la modifica di uno dei sensori. Questa classe fornisce un metodo *onSensorChanged*, il quale rileva il cambiamento relativo ai dati dei sensori permettendoci di rilevarli ed inserirli in un *array* creato in precedenza al fine di poter registrare i dati presi in considerazione. Le registrazioni vengono effettuate con una frequenza di 10 Hz, cioè avviene ogni qualvolta viene rilevata una modifica per uno dei valori dei sensori considerati.

Il frammento di codice sottostante mostra questo processo, considerato nel solo caso dell'accelerometro:

```
public void onSensorChanged(SensorEvent event){  
    if(event.sensor.getType() == Sensor.TYPE_ACCELEROMETER){  
  
        System.arraycopy(  
            event.values, 0, SensorVal, 0, event.values.length);  
        System.arraycopy(  
            event.values, 0, mLastAcc, 0, event.values.length);  
        mLastAccSet=true;  
    }  
}
```

Ogni qualvolta viene chiamato il metodo *onSensorChanged*, viene anche effettuata una chiamata al metodo *Write* relativo alla classe *CSV*. Il metodo

*Write* riceve in input i dati relativi ai sensori, la variabile relativa alla sezione di schermo cliccata e il *path* relativo al *file* precedentemente creato all'interno della memoria del dispositivo. A questo punto, il metodo *Write* aggiorna il *file*, aggiungendo una riga nella seguente forma:

$$\Theta(t) = \langle t, A_{x,y,z}, M_{x,y,z}, O_{x,y,z}, \Omega \rangle$$

In cui  $t$  corrisponde all'istante in millisecondi in cui i valori dei sensori sono stati modificati e in cui i valori  $A$ ,  $M$  e  $O$ , relativi ai vari assi, corrispondono rispettivamente ad accelerometro, magnetometro e orientamento. Infine,  $\Omega$  corrisponde alla sezione di schermo su cui si effettua il *touch* in quell'istante. Nel caso in cui non venga selezionata nessuna porzione di schermo, il suo valore é -1. Questa serie di valori registrati in relazione al momento della loro rilevazione viene definito *serie storica*. Una serie storica, quindi, non é altro che una serie di valori variabili ordinati rispetto al tempo, con lo scopo di studiare l'andamento di uno specifico fenomeno nel tempo. Nella figura 2.4 relativa al solo caso dell'accelerometro, viene rappresentata la serie storica dei valori ottenuti da una sessione di registrazione, in cui viene effettuato un *touch event* (valore indicato tra due linee verticali rosse) per ogni sezione dello schermo. L'immagine sottostante é relativa al caso in cui lo schermo sia diviso in 8 sezioni.

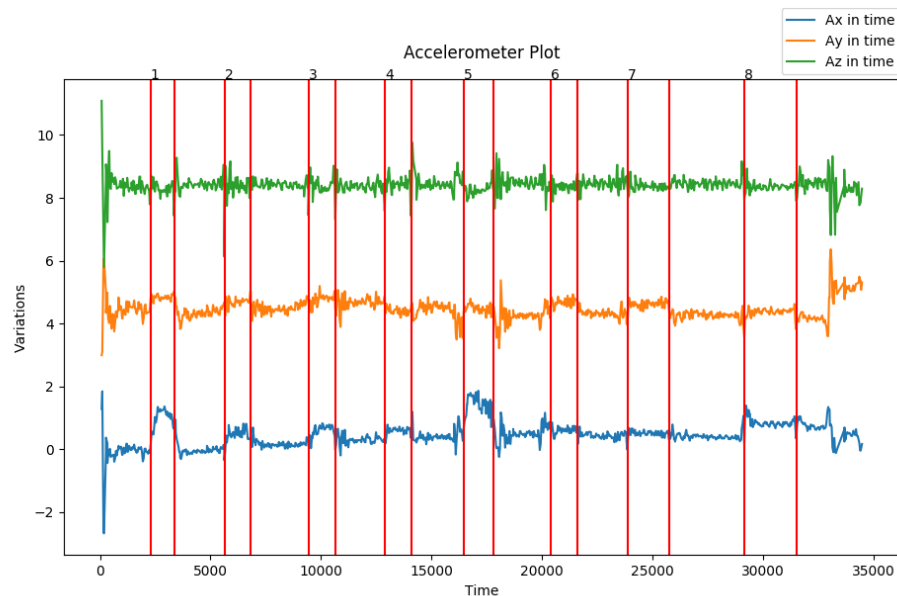


Figura 2.4: Dati grezzi Accelerometro per i tre assi nel tempo, 1 click per sezione nel caso di 8 sezioni.

Se l'applicazione dovesse essere messa in pausa, terminata in maniera involontaria o se, piú semplicemente, si dovesse scegliere di terminare la registrazione attuale cliccando sul tasto *Finish* presente nel menú, viene effettuata una richiesta *FTP* asincrona dal dispositivo verso un *server*. Questa richiesta viene usata al fine di inviare ed eliminare dal dispositivo il *file* appena generato, al fine di poterlo successivamente utilizzare per l'analisi dei dati. Inoltre, questa richiesta viene effettuata chiamando la classe *UploadFile*, estensione di *AsyncTask*, la quale ha il compito di eseguire dei *task* in background senza modificare l'UI, notificando al termine l'avvenuta operazione.



## Capitolo 3

### Modello

La seconda fase del progetto corrisponde alla fase di analisi dei dati ottenuti dal dispositivo. Il linguaggio di programmazione usato per tale fase é *Python*: tale scelta é giustificata dal fatto che la libreria utilizzata per implementare il modello di *machine-learning* é scritta in *Python*. Tale libreria é *Scikit-learn*, la quale fornisce gli strumenti ideali per poter creare e addestrare un modello di *machine-learning*.

Per rendere possibile tale analisi vengono calcolate delle *features* a partire dai dati grezzi relativi all'accelerometro e al magnetometro e da una separazione di tutti i *touch events* che interessano il dispositivo in questione, in modo da poterli identificare chiaramente. Infine, queste *features* possono essere date in pasto ad un modello di *machine-learning*, il quale le utilizzerá al fine di effettuare delle predizioni sulle sezioni di schermo selezionate. Dopo questa breve spiegazione generale, ci si puó concentrare piú in dettaglio nella descrizione di questo procedimento. Durante questa descrizione, verranno mostrati i grafici relativi all'isolamento dei *touch events* e relativi alle *features* solo per il caso in cui lo schermo é diviso in 8 sezioni per una semplicitá di lettura.

## 3.1 Touch events e calcolo delle features

### 3.1.1 Isolare i touch events

La prima operazione necessaria al fine di calcolare le *features* indispensabili al modello é quella di separare ogni *touch event* all'interno del *file CSV* contenente i dati grezzi. In questo modo viene cosí generato un nuovo file privo del "rumore" dovuto alla fase di transizione tra un evento *touch* e l'altro. Nel considerare ogni evento *touch*, tuttavia, non viene registrato solo l'intervallo in cui viene mantenuta la pressione del dito sullo schermo, ma anche i valori relativi ai 200 millisecondi prima e dopo l'avvenimento dell'azione di *touch* sullo schermo. Questi intervalli vengono considerati allo scopo di mantenere in memoria i dati relativi al momento in cui l'utente sta per premere la sezione di schermo considerata e il momento successivo alla pressione sullo schermo. Tutto ciò risulta particolarmente utile in quanto permette di tenere traccia anche dell'inclinazione a cui é sottoposto il dispositivo, quando si passa dal non selezionamento delle sezioni alla selezione e viceversa. In figura 3.1 viene mostrato il grafico risultante dalla selezione dei *touch events*, a partire dai dati evidenziati dal grafico in Figura 2.4.

### 3.1.2 Calcolo delle features

Dal momento in cui, a partire dai dati grezzi, vengono identificati i *touch events*, risulterà possibile calcolare le *features* che verranno poi utilizzate dal modello di *machine-learning* per il riconoscimento delle sezioni interessate. Nel contesto dei modelli di machine learning, una *feature* può essere considerata come una proprietà misurabile, in generale, a partire da un ben determinato fenomeno osservato. Quindi, risulta di fondamentale importanza la scelta di tali *features*, al fine di descrivere nel modo più accurato possibile il fenomeno studiato.

Nel caso presentato all'interno di questo elaborato, le *features* verranno calcolate con una frequenza di 20 Hz in relazione ai dati registrati dai sensori



considerati fino ad ora. Le features scelte per rappresentare il fenomeno di cui ci si interessa in questo elaborato sono:

- $Max_{x,y,z}^{A,M,0}$ : corrispondente al massimo valore dell'intervallo considerato;
- $Min_{x,y,z}^{A,M,0}$ : corrispondente al minimo valore dell'intervallo considerato;
- $Max - Min_{x,y,z}^{A,M,0}$ : corrispondente alla differenza tra il massimo e il minimo valore dell'intervallo considerato;
- $Avg_{x,y,z}^{A,M,0}$ : corrispondente alla media tra i valori dell'intervallo considerato;
- $Std_{x,y,z}^{A,M,0}$ : corrispondete alla deviazione standard tra i valori dell'intervallo considerato.

Ogni *feature* viene calcolata per ogni sensore considerando i suoi valori lungo gli assi: x,y e z. In Figura 3.2 viene mostrato il grafico dei valori delle *features* relative alla media (*Avg*) nel caso in cui si considerino i dati registrati dal solo accelerometro. Inoltre, si fa riferimento al caso in cui viene eseguito un singolo *touch event* per sezione e in cui lo schermo é diviso in 8 sezioni. Come si può notare da una lettura preliminare dei dati a partire dal grafico ottenuto, vi sono dei picchi in corrispondenza dei *touch events* relativi alle sezioni 1 e 5. Tali *touch events* mostrano un comportamento che potrebbe essere utilizzato come *pattern* per il riconoscimento delle sezioni da parte del modello di *machine-learning*.



Figura 3.1: Eventi touch Accelerometro, 1 click per sezione nel caso di 8 sezioni.

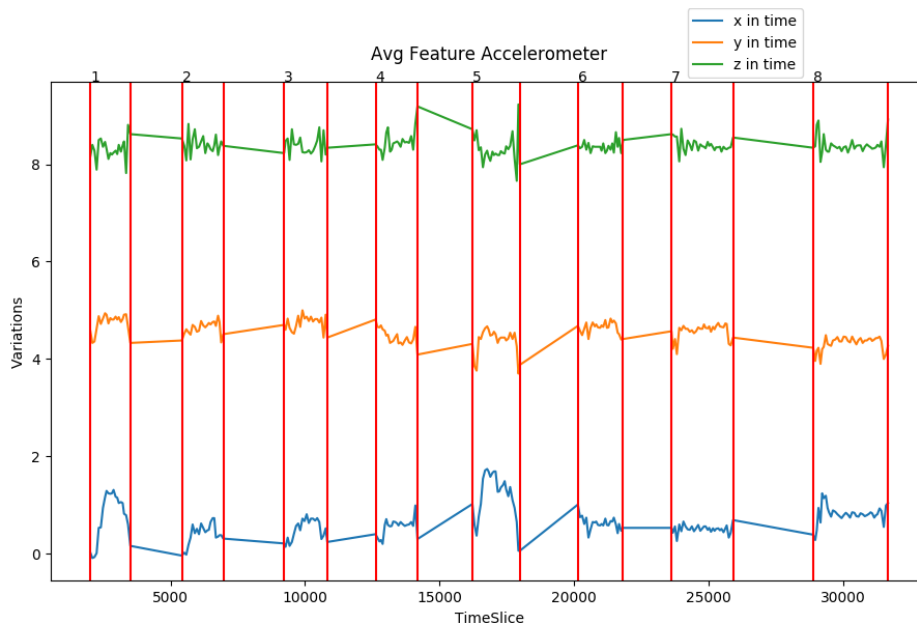


Figura 3.2: Variazione media(Avg) nel tempo del caso mostrato in Figura 3.1.

## 3.2 Modello

### 3.2.1 Background

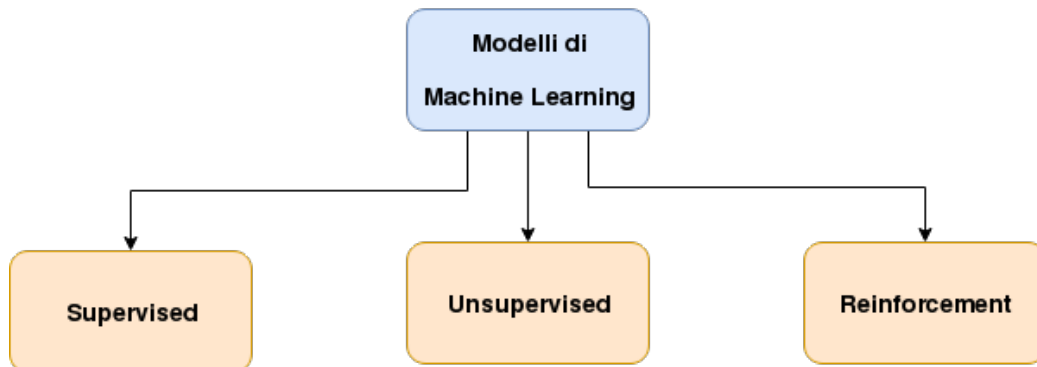


Figura 3.3: Categorie dei modelli di *Machine-learning*.

Dopo aver calcolato le *features* relative ai sensori, come descritto nella sezione 3.1.2, è possibile addestrare un modello di *machine-learning* al riconoscimento delle sezioni di schermo selezionate. In generale, un modello di *machine-learning* ha l'obiettivo di svolgere nuovi compiti o riconoscere nuovi dati. Queste operazioni possono aver luogo solo dopo che il modello ha avuto modo di apprendere il fenomeno studiato a partire da dati ad esso relativi. In tal modo, il modello può dare avvio ad una fase di addestramento indispensabile al riconoscimento dei dati di partenza.

Si possono distinguere tre principali categorie di metodi per la soluzione di problemi relativi ai modelli di *machine-learning*. Queste tre categorie, così come riportate in Figura 3.3, possono essere di vari tipi:

- **Supervised:** il modello viene addestrato a riconoscere dei *Target* tramite una serie di dati di input chiamati, non a caso, *Training Data*;
- **Unsupervised:** contrariamente a ciò che avviene nel caso del *Supervised*, il modello riceve solo una serie di dati in *input* senza nessuna associazione ai *Target* da riconoscere. In questo caso, è compito dell'algoritmo riconoscere un *pattern* al fine di trarne dei risultati;

- **Reinforcement**: il modello si adatta ai dati ricevuti in *input* tramite un sistema di "ricompense" dette rinforzo.

Nel contesto di questo elaborato, considereremo il caso in cui il modello di *machine-learning* sia di tipo *Supervised*. Il *Supervised machine-learning* é una tecnica di *machine-learning* che ha lo scopo di istruire un modello con l'obiettivo di effettuare delle previsioni in maniera automatica sui valori ricevuti in *input*, i quali rappresentano un ben determinato fenomeno. Al fine di raggiungere tale scopo, il modello si basa su una serie di esempi che gli vengono precedentemente forniti. Questi modelli verranno utilizzati come *Training dataset* cosí da permettere al modello di *machine-learning* di riconoscere il fenomeno descritto a partire dai dati ricevuti in *input*. Nel caso specifico di questo elaborato, una parte dei dati che il modello riceverá in *input* verrá utilizzata come *Training Dataset* necessario al fine di addestrare il modello. La parte restante dei dati ricevuti in *input* verrá utilizzata dal modello per tentare delle predizioni sulla percentuale di riconoscimento delle sezioni di schermo selezionate durante la registrazione.

### 3.2.2 Implementazione del modello e casi considerati

Come già anticipato nella sezione 3.2.1, il modello implementato all'interno di questo elaborato é un modello di *supervised machine-learning*, il quale sfrutterá i dataset ricevuti in *input* al fine di ottenere un *Training dataset* corrispondente al 60 % dei dati. Il restante 40 % verrá invece utilizzato al fine di effettuare delle predizioni sulle sezioni di schermo selezionate.

Le *features* considerate dal modello per effettuare tali predizioni sono quelle a cui si é accennato all'interno della sezione 3.1.2. Inoltre, ad ogni record relativo alle *features* sará anche associato il rispettivo  $\Omega$  corrispondente alla sezione di schermo selezionata all'interno di quel determinato record. L' $\Omega$  ci permetterà di distinguere ogni sezione di schermo relativa al record, sia in fase di training del modello sia durante i test relativi ai dati ricevuti.

Quindi, dopo la memorizzazione dei dati all'interno di un *array* contenente le *features* e di un altro *array* contenente i *target* relativi alle sezioni di

schermo, il modello di *machine-learning* elimina una serie di record da 0 a 10, relativi all'inizio e alla fine di ogni *touch event* precedentemente isolato. In tal modo, é possibile minimizzare il piú possibile il "rumore" ricevuto in *input* dai dati. Un'ultima importante scelta implementativa riguarda l'uso del metodo di apprendimento in fase di *training*. Questo metodo é rappresentato da un algoritmo chiamato *Random decision forest*, il quale si occupa di costruire una serie di alberi di decisione a partire dai dati ricevuti e di generare predizioni per ogni albero creato. In questo modo é possibile evitare un *overfitting* dei dati, poiché in presenza di questo fenomeno, causato dalla presenza di un numero eccessivo di dati rispetto al numero di osservazioni, i risultati si riveleranno piú incerti.

Successivamente, quando lo script termina la sua esecuzione, il modello di *machine-learning* genera un *file CSV* in cui ogni riga si presenta nella seguente forma:

$$\Pi(\rho) = \langle \rho, Accuracy, Std \rangle$$

In cui  $\rho$  rappresenta il numero di record eliminato da ogni serie di valori relative a un *touch event*; *Accuracy* rappresenta la precisione secondo cui il modello é in grado di predire le sezioni di schermo selezionate sul dispositivo; *Std* la deviazione standard relativa alla predizione. Ai fini del progetto presentato all'interno di questo elaborato sono stati creati due modelli di *machine-learning* equivalenti al modello descritto precedentemente all'interno di questa sezione. L'unica differenza tra i due modelli consiste nel fatto che, mentre il primo riceverá in *input* i dati relativi unicamente alle *features* dell'accelerometro, il secondo riceverá in input sia le *features* relative all'accelerometro sia quelle relative al magnetometro. Il secondo modello di *machine-learning* ha lo scopo di migliorare le predizioni sulle sezioni di schermo cliccate. Questo miglioramento puó avvenire grazie all'utilizzo dei dati del magnetometro, i quali permettono di ottenere una rappresentazione piú dettagliata dell'inclinazione del dispositivo dovuta al *touch event*. Dall'insieme di tutte queste operazioni viene escluso l'utilizzo del sensore relativo all'o-

rientamento del dispositivo considerato, poiché i suoi valori sono interamente calcolati a partire dai dati relativi all'accelerometro e al magnetometro. Dopo aver ottenuto diversi *datasets* di registrazioni relative ai sensori e alle varie possibilità di divisione dello schermo in sezioni, ci si è potuti concentrare sullo studio di un ulteriore caso, simile a quello in cui un'applicazione malevola riesca ad ottenere una serie di registrazioni non consecutive nel tempo. Tuttavia, al fine di effettuare delle analisi su tali registrazioni non consecutive, è necessaria una fase preliminare a quelle descritte fino ad ora, in cui è indispensabile un ricalcolo dei dati grezzi. Questo ricalcolo viene effettuato in modo da studiare i dati non secondo il loro valore assoluto, ma in modo tale da considerarli come variazioni rispetto ad un *record* iniziale. Quindi, ogni *record* viene ricalcolato come risultato della differenza tra  $a_{x,y,z}$ , il quale rappresenta il caso fissato per l'esempio dell'accelerometro, e  $P_{x,y,z}^A$ , il quale rappresenta i valori dei vari *record* all'interno del *dataset* grezzo. La stessa procedura viene, ovviamente, eseguita anche sui dati relativi al magnetometro. Quest'ultima operazione produrrà un nuovo *file*, in cui ogni campo dei *record* viene ricalcolato come segue:

$$\Theta'(t) = \langle t, a_{x,y,z} - P_{x,y,z}^A, m_{x,y,z} - P_{x,y,z}^M \rangle$$

Quindi, dopo aver ricalcolato ogni registrazione, risulta possibile unire i *datasets* da studiare all'interno di un unico *file*, al fine di poter eseguire le operazioni necessarie per calcolare le *features* già descritte in 3.1.2. Infine, sarà possibile dare in pasto tali dati ai modelli di *machine-learning* descritti all'interno di questa sezione.

Nel prossimo capitolo verranno trattati in dettaglio i risultati ottenuti tramite questo processo.

# Capitolo 4

## Risultati

La terza fase di questo progetto consiste in un'applicazione di natura pratica dei risultati ottenuti: partendo dalle percentuali di riconoscimento ottenute dai modelli precedentemente descritti, sarà infatti possibile costruire un albero delle possibili parole digitate dall'utente. Prima di descrivere nel dettaglio questa fase, sarà tuttavia opportuno concentrarsi sulla descrizione dei risultati ottenuti dai modelli di *machine-learning*.

Ad esempio, i grafici mostrati di seguito sono calcolati a partire dalle *features* relative agli esempi mostrati in precedenza. Inoltre, i grafici citati fanno riferimento al solo al caso in cui vi sia una divisione dello schermo in 8 sezioni e in cui venga effettuato un *touch event* per sezione. Infine, tali grafici avranno tutti come variabile per l'asse Y l'*Accuracy*, cioè la precisione secondo cui il modello è in grado di effettuare le predizioni sulle sezioni di schermo selezionate. Come variabile per l'asse X sarà, invece, assegnato il *Bound*, cioè il numero di *record* eliminati all'inizio e alla fine di ogni *touch event* al fine di minimizzare il "rumore" e di rendere più precise le predizioni. Sulla base di queste considerazioni, è anche possibile notare come nei grafici siano presenti delle frecce verticali in corrispondenza dei punti in cui viene rappresentata la relazione tra il *Bound* e l'*Accuracy*: tali frecce rappresentano il valore della *Std*, il quale delinea il grado di incertezza delle predizioni ottenute.

## 4.1 Risultati relativi alle predizioni effettuate dalle *features* dell'accelerometro

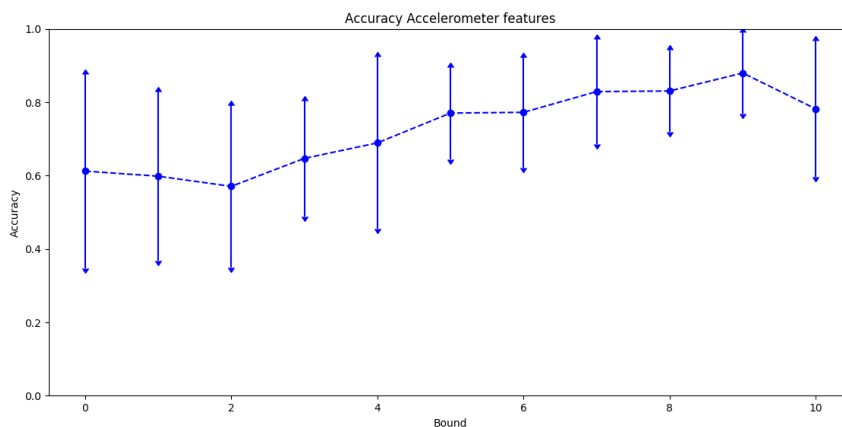


Figura 4.1: Grafico relativo all'accuratezza delle predizioni per 8 sezioni, nel caso in cui vengano considerate le *features* dell'accelerometro inerenti ad una singola registrazione.

Il caso del grafico mostrato in Figura 4.1 é il caso in cui il modello effettua delle predizioni secondo il metodo descritto nella sezione 3.2.2. Queste predizioni vengono effettuate a partire da un *file* contenente una singola sessione di registrazione dei dati relativi all'accelerometro. A partire dall'analisi e dalla studio di questi dati, vengono calcolate le *features* che verranno successivamente date in pasto al modello di *machine-learning*. Da un'analisi dei primi risultati, in cui il *Bound* é compreso tra 0 e 2, si puó notare come questi siano sicuramente quelli caratterizzati da un'accuratezza minore, causata dall'incertezza dovuta alla presenza di dati superflui relativi ai *touch events*. Infatti, aumentando il valore relativo ai *records* da eliminare al fine di isolare l'evento *touch*, si puó notare un chiaro aumento nella precisione delle predizioni, la quale ha il massimo valore per *Bound* uguale a 9. In questo caso specifico, sará possibile notare anche una chiara diminuzione della conseguente *Std*.



## 4.2 Risultati relativi alle predizioni effettuate dalle *features* di accelerometro e magnetometro

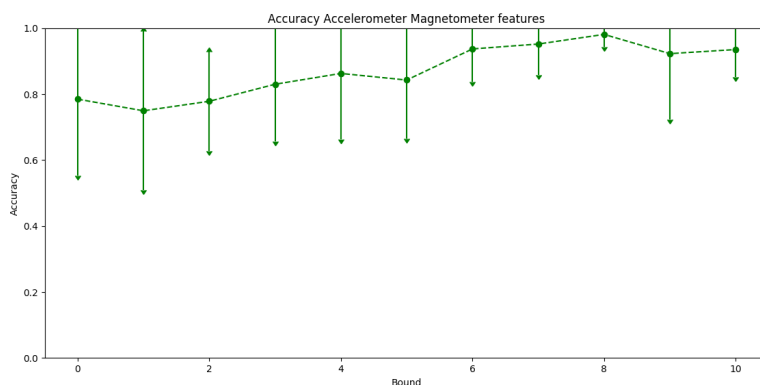


Figura 4.2: Grafico relativo all'accuratezza delle predizioni per 8 sezioni, a partire dai dati relativi ad accelerometro e magnetometro all'interno di una singola registrazione.

Questo elaborato, come già specificato all'interno dell'*Introduzione*, non tratterà l'analisi dei soli dati relativi all'accelerometro, ma anche dei dati ottenuti dalla registrazione del magnetometro, al fine di ottenere una maggiore precisione nel riconoscimento dell'inclinazione del dispositivo. Il progetto su cui si basa la trattazione contenuta all'interno di questo elaborato, inoltre, è stato sviluppato anche al fine di sfruttare i dati relativi al sensore giroscopio. Tuttavia, tali dati non verranno trattati nel contesto proprio dell'elaborato stesso, in quanto il dispositivo su cui sono state effettuate le registrazioni è sprovvisto di tale sensore. Detto ciò, in Figura 4.2 viene mostrato il risultato ottenuto dalle predizioni relative al modello di *machine-learning* addestrato a partire da *features* calcolate considerando i dati relativi a magnetometro e accelerometro. Sulla base di queste considerazioni, si può chiaramente notare come il modello di *machine-learning* sia effettivamente in grado di riconoscere

le sezioni di schermo cliccate con una maggiore accuratezza. In particolare, il grafico mostra un andamento simile a quello mostrato in Figura 4.1, ma con un'accuratezza migliorata all'incirca del 20%, la quale può inoltre raggiungere un picco massimo dell'88% in corrispondenza di un *Bound* con valore 8.

### 4.3 Risultati relativi al ricalcolo dei dati grezzi

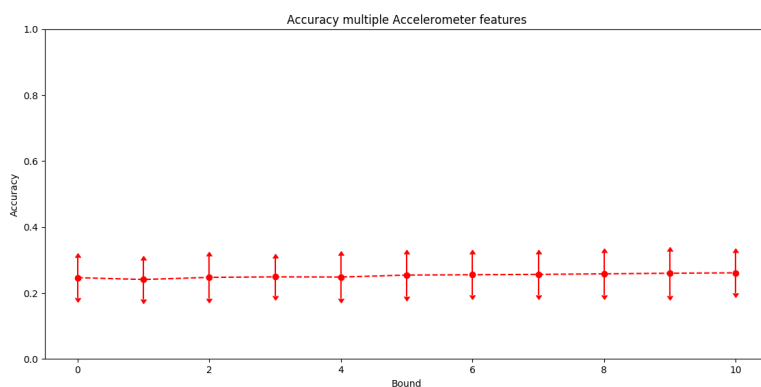


Figura 4.3: Grafico sull'accuratezza delle predizioni per 8 sezioni di schermo inerenti ai dati grezzi dell'accelerometro nell'arco di 10 registrazioni.

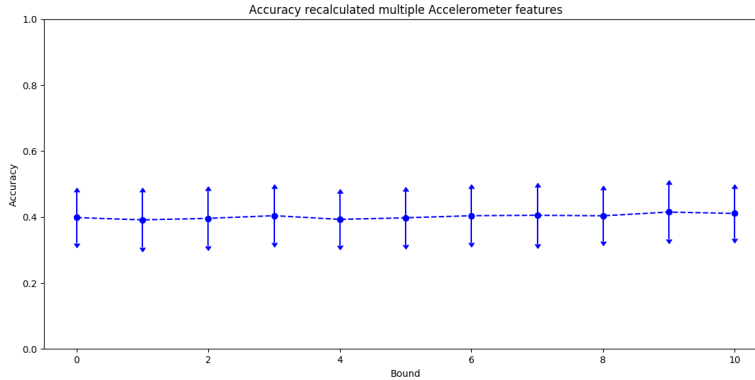


Figura 4.4: Grafico sull'accuratezza delle predizioni per 8 sezioni inerenti ai dati ricalcolati dall'accelerometro nell'arco di 10 registrazioni.

Come già anticipato nella sezione 3.2.2, oltre al caso in cui venga effettuata da parte dei sensori una singola registrazione sul dispositivo, viene anche preso in considerazione il caso in cui vengano effettuate da parte dei sensori più registrazioni in momenti diversi. Quindi, viene considerato anche il caso in cui vengano scelti orientamenti diversi da parte dell'utente. Nel caso specifico presentato all'interno delle Figure 4.3 e 4.4 verranno presi in considerazione 10 *file* diversi contenenti le registrazioni dei sensori nel caso in cui lo schermo del dispositivo sia diviso in 8 sezioni e siano stati eseguiti 5 *touch event* per sezione. Tale operazione viene effettuata in modo tale da ottenere una maggiore quantità di dati da utilizzare in fase di addestramento del modello di *machine-learning*. I due grafici mostrati rappresentano, inoltre, l'accuratezza relativa al solo caso in cui il modello si dedichi all'analisi dei dati relativi alle *features* dell'accelerometro. Lo scopo di tali grafici è quello di mostrare come sia effettivamente possibile ottenere un netto miglioramento nel riconoscimento delle sezioni da parte del modello di *machine-learning* nel caso in cui avvenga un ricalcolo dei dati grezzi, così come mostrato anche nella sezione 3.2.2. Infatti, è possibile notare come tra la figura 4.3 e la figura 4.4 vi sia un miglioramento nell'accuratezza delle predizioni pari a circa il 20%. Inoltre, è anche possibile notare come, a differenza dei grafici presentati all'inizio di questo capitolo, in cui viene considerata solo una singola regi-

strazione, l'andamento del grafico risulta piú costante al variare dei valori di *Bound*.

#### 4.4 Risultati relativi alle predizioni delle *features* ottenute dal ricalcolo di accelerometro e magnetometro

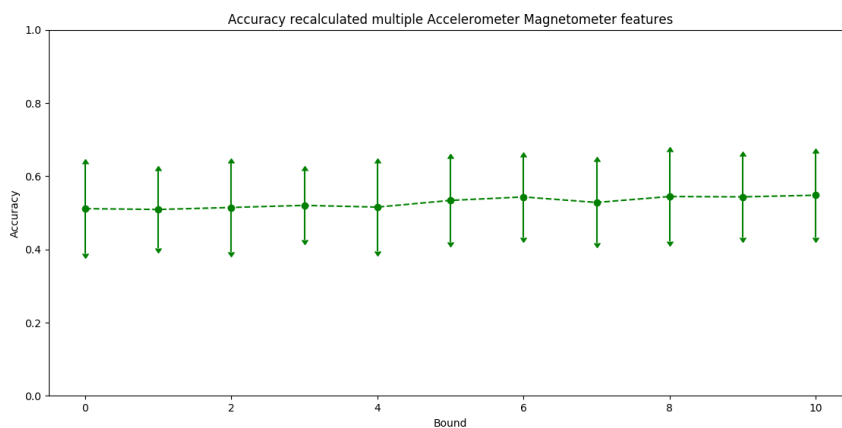


Figura 4.5: Grafico relativo all'accuratezza delle predizioni per 8 sezioni inerenti alle *features* di accelerometro e magnetometro ed effettuato a partire dai dati ricalcolati nell'arco di 10 registrazioni.

Dopo aver confermato la possibilità di migliorare le predizioni nel caso in cui vengano considerate piú registrazioni contemporaneamente, cosí come mostrato nella sezione precedente, viene eseguito lo stesso procedimento usato per l'accelerometro in sezione 4.3 al fine di effettuare il ricalcolo dei dati relativi al magnetometro. Sulla base di queste operazioni, risulta quindi possibile dare in pasto le *features* relative a magnetometro e accelerometro al modello di *machine-learning*.

In Figura 4.5 viene mostrato il risultato ottenuto a partire dalle *features* cal-

colate tramite il metodo descritto all'inizio di questa sezione. Da un'attenta lettura del grafico in questione si può notare, in particolare.

- Un chiaro miglioramento nelle percentuali relative alle predizioni. Tale miglioramento è chiaramente dovuto all'aggiunta delle *features* relative al magnetometro;
- Un andamento costante simile a quello mostrato nelle Figure 4.3 e 4.4.

## 4.5 Risultati per i casi: 8, 16, 32, KBD

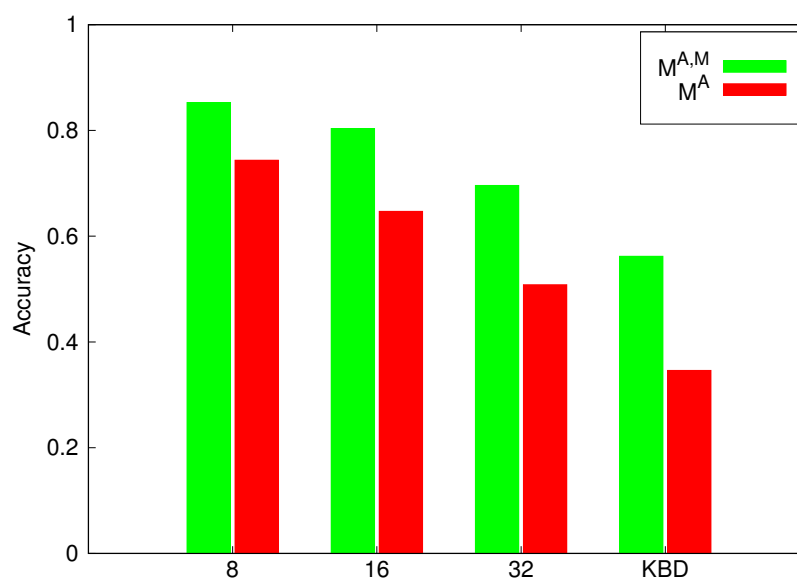


Figura 4.6: Grafico a barre relativo ai migliori risultati ottenuti dai dati di una registrazione, nel caso in cui si consideri solo l'accelerometro *in rosso* e il caso in cui venga aggiunto anche il magnetometro *in verde*.

All'interno del Capitolo 2 viene mostrata la possibilità di sezionare l'area relativa allo schermo del dispositivo in più modi prima di effettuare una registrazione. All'interno di questa sezione vengono evidenziati i casi più interessanti relativi proprio a questa operazione e inerenti ad una divisione dello schermo in: 8, 16, 32 e KBD sezioni, dove KBD rappresenta il caso in

cui le porzioni di schermo premute siano quelle relative alla parte inferiore dello schermo del dispositivo considerato. Questo caso viene preso in considerazione al fine di emulare il *touch* sulla tastiera virtuale del dispositivo.

La Figura 4.6 mostra i migliori risultati ottenuti e inerenti ai casi considerati all'inizio di questa sezione. In particolare, viene considerato il caso in cui si abbia una sola registrazione relativa alla divisione dello schermo in: 8, 16, 32 e KBD sezioni. Sulla base di un'attenta analisi del grafico contenuto nella figura citata, risulta evidente, come già evidenziato all'interno della sezione 3.2.2, l'ottenimento di un'accuratezza maggiore nel caso in cui ci si basi sulla presenza delle *features* calcolate a partire da accelerometro e magnetometro (Barre verdi in Figura 4.6), rispetto al caso in cui venga considerata la sola presenza delle *features* dell'accelerometro (Barre rosse in Figura 4.6). É ulteriormente possibile notare come il caso KBD possieda un'accuratezza inferiore, poiché, in questo caso specifico, ogni sezione di schermo é 4 volte inferiore rispetto alla precedente, rendendo così nettamente più difficile il riconoscimento della sezione selezionata.

Infine, é interessante mostrare i risultati ottenuti sui medesimi casi considerati in Figura 4.6, considerandoli però non più in riferimento ai dati relativi ad una singola registrazione sul dispositivo, ma ai dati relativi a 10 registrazioni differenti. Tali dati, inoltre, vengono calcolati tramite il metodo descritto nella sezione 4.3.

Questo caso specifico viene mostrato in Figura 4.7. Sulla base della lettura del grafico contenuto nella figura appena citata, é possibile notare come vengano ottenute delle percentuali inferiori rispetto al caso in cui vengano considerate registrazioni singole sui sensori. Eppure, nonostante i risultati ottenuti siano inferiori rispetto a quelli considerati nel caso precedente, si ottengono comunque dei risultati migliori rispetto al caso in cui si consideri la presenza dei dati relativi al solo accelerometro. Da questi risultati appare, inoltre, evidente come la percentuale di riconoscimento dei tasti nel caso di *KBD* abbia un valore pari a circa il 40%, a partire dallo studio di diverse registrazioni, e un valore di circa il 60%, a partire dallo studio di una sola

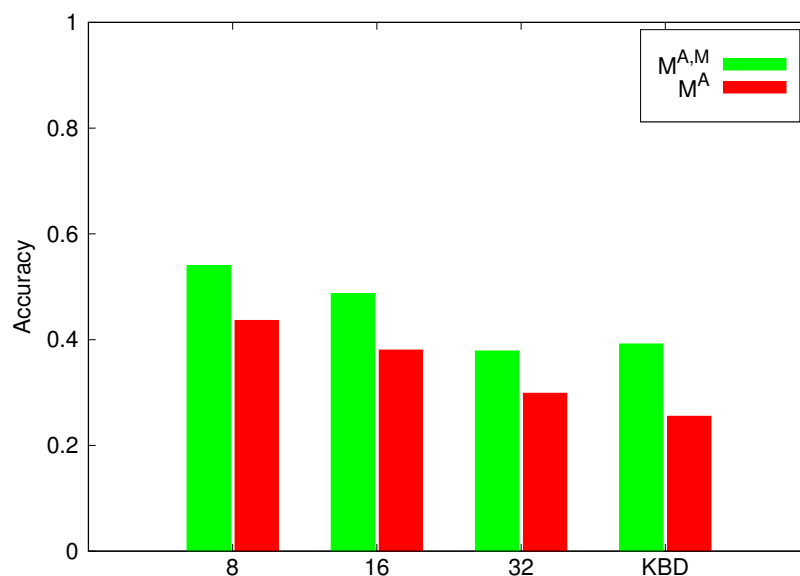


Figura 4.7: Grafico a barre relativo ai migliori risultati ottenuti e inerenti ai dati di 10 registrazioni, nel caso in cui venga considerato solo l'accelerometro *in rosso* e nel caso in cui si consideri anche il magnetometro *in verde*.

registrazione. A questo punto, sar  possibile utilizzare tali risultati al fine di mostrare la reale possibilit  di comprendere le parole digitate dall'utente su una tastiera virtuale.

## 4.6 Riconoscimento delle parole digitate

Dopo aver ottenuto i risultati utili al riconoscimento dei tasti digitati sulla tastiera virtuale del dispositivo da parte dell'utente,   possibile testare l'effettiva possibilit  di riconoscere le parole digitate. Al fine di rendere possibile il riconoscimento delle parole,   necessario correlare tutte le sequenze di caratteri digitati dall'utente e costruire il relativo albero dei possibili caratteri digitati. In questo modo, si potr  ottenere la possibilit  di classificare la parola in base alla probabilit  associata ad ogni sua lettera. In fase di progettazione  , tuttavia, risultato evidente come la costruzione dell'albero delle possibilit , soprattutto per parole pi  lunghe, possa rallentare l'esecuzione

dello *script*, portando anche alla creazione di casi di parole inesistenti. Quindi, al fine di ovviare a questo problema, si é deciso di utilizzare un dizionario inglese composto da oltre 470.000 parole: sará cosí possibile confrontare la parola costruita fino al nodo considerato. Nel caso in cui la parola trovata non appartenga al dizionario considerato, viene interrotto il ramo, il quale viene anche escluso dalle possibili parole finali.

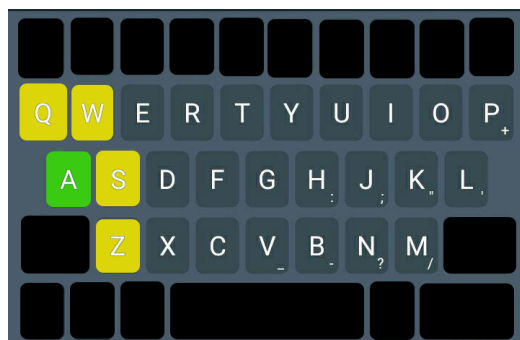


Figura 4.8: Esempio in cui  $C_i = a$  e  $A(C_i) = q, w, s, z$ .

L'albero viene costruito in modo tale da definire, per ogni carattere digitato alla  $i$ -esima posizione e indicato come  $C_i$ , una lista  $A(C_i)$  di tutti i caratteri adiacenti a quello considerato. Tale costruzione é mostrata nell'esempio in Figura 4.8, in cui il carattere  $C_i$  é  $a$  e i cui adiacenti  $A(C_i)$  sono:  $q, w, s$  e  $z$ . Inoltre, al fine di aggiungere un certo grado di incertezza, si procede all'associazione ad ogni carattere ad una ben determinata probabilitá. Tale associazione viene effettuata perché non é possibile ottenere un riconoscimento certo del carattere digitato.

Nel caso del carattere corrente  $C_i$ , tale probabilitá sará  $p(C_i)$  calcolata come segue:

$$p(C_i) = 1 - (|A(C_i)|) * p_c(A(C_i))$$

in cui  $|A(C_i)|$  é il numero dei caratteri adiacenti a  $C_i$  e  $p_c(A(C_i))$  corrisponde alla probabilitá che il carattere  $c$  faccia parte degli adiacenti  $A(C_i)$ . Tale



probabilità viene calcolata nel modo che segue:

$$p_c(A(C_i)) = \frac{1}{|A(C_i)| + 2}$$

L'idea espressa da questo metodo di calcolo risiede nel fatto che il carattere  $C_i$  viene considerato con una probabilità maggiore rispetto ai suoi adiacenti. Questa operazione permetterà di esplorare e classificare tutte le varie parole possibili.

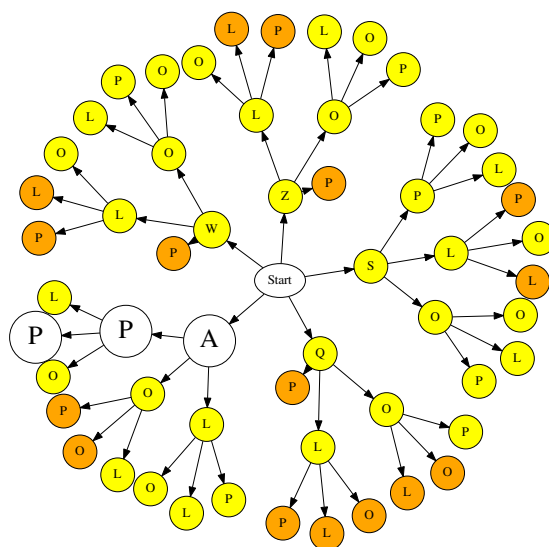


Figura 4.9: Albero delle possibilità per la parola *App*.

A questo punto è possibile affermare che, durante la costruzione dell'albero delle possibili parole, ad ogni livello  $i$  viene inserito il nodo  $C_i$  corrispondente al carattere digitato e ai suoi adiacenti  $|A(C_i)|$ . Inoltre, viene associata una probabilità ad ogni transizione dal livello  $i - 1$  al livello  $i$  pari a:  $p(C_i)$ , nel caso in cui la transizione porti a  $C_i$ , o pari a  $p_c(A(C_i))$ , nel caso in cui la transizione porti a un nodo relativo agli adiacenti di  $C_i$ . Infine, dopo la creazione di ogni livello, viene effettuato un controllo relativo alla parola formata. Tale controllo considera i caratteri presenti in un cammino dal livello 0 fino alla foglia attuale. Solo nel caso in cui all'interno del dizionario

non vi sia nessuna parola che inizia con la porzione stringa costruita scorrendo i valori dell'albero, il nodo contestualmente considerato viene eliminato dall'albero. Così facendo, viene interrotto il relativo ramo generato, al fine di rendere piú efficiente la costruzione dell'albero delle parole possibili.

Un esempio di costruzione di tale albero viene presentato in Figura 4.9. In tale esempio viene considerata per comoditá la parola *app*, in quanto per parole piú lunghe il conseguente albero risulterebbe poco leggibile. Al livello 0 viene presentato il carattere *a* colorato in bianco, il quale corrisponde a  $C_i$ , mentre i suoi adiacenti, ovvero  $A(C_i) = q, w, s, z$ , sono colorati in giallo. Successivamente, al livello 1, viene presentato il carattere *p*, in qualitá  $C_i$ , e i caratteri *o, l*, in qualitá di suoi adiacenti. Tale procedimento viene ripetuto anche in corrispondenza del livello 2. Sulla base di tali considerazioni, é possibile notare come, giá in corrispondenza del livello 1 dell'albero, i caratteri colorati in arancione corrispondano ai rami che conducono all'individuazione di parole inesistenti all'interno del dizionario considerato. Tale operazione conduce cosí anche all'eliminazione del loro relativo ramo, al fine di permettere una piú efficiente generazione e ricerca delle parole possibili. Quando, infine, si ottiene la creazione dell'albero risultante dalla parola *app*, é possibile notare la presenza, in bianco, dei nodi che compongono la parola con probabilitá maggiore e, in giallo, delle altre possibili parole caratterizzate da una probabilitá inferiore. Inoltre, é anche importante notare che in corrispondenza di parole di pochi caratteri si ottengono diverse foglie e, quindi, piú combinazioni di parole da verificare. Invece, quando il numero dei caratteri aumenta notevolmente, vi é una diminuzione delle foglie presenti e, quindi, delle parole da sottoporre effettivamente ad un controllo. Tale operazione permette, perció, di ottenere risultati migliori per il riconoscimento delle parole prese in considerazione.

## 4.7 Risultati relativi al dizionario

Dopo aver generato l'albero delle possibilità con le relative probabilità per ogni parola, risulta possibile studiare la correttezza delle parole trovate a partire da quella esatta, tramite un algoritmo chiamato *distanza di Levenshtein*. Questa procedura corrisponde alla ricerca del minimo numero di cambiamenti a cui sottoporre una stringa A al fine di renderla uguale ad una stringa B. Quando si parla di cambiamenti si intende:

- La sostituzione di un carattere, come nel caso in cui la parola A sia Rane e la parola B sia Pane. La sostituzione corrisponde alla modifica del carattere R in P per la parola A;
- La rimozione di un carattere, come nel caso in cui la parola A sia Principio e la parola B sia Principi. La rimozione consiste nell'eliminazione del carattere o in A;
- L'aggiunta di un carattere, come nel caso in cui la parola A sia Principi e la B sia Principio. L'aggiunta, caso ovviamente opposto rispetto alla rimozione, consisterà nel concatenare il carattere o alla stringa A.

Un esempio dell'applicazione di questi cambiamenti può essere mostrato dall'esecuzione di un confronto tra la parola A *Cane* e la parola B *Mano*. Il valore di ritorno, in questo caso, sarà 2, poiché deve essere effettuata la modifica del carattere c in m e il carattere e in o, al fine di ottenere la stringa B a partire dalla stringa A. Grazie all'utilizzo della *distanza di Levenshtein* e alla presenza della probabilità associata ad ogni parola ottenuta dall'albero delle parole possibili, sarà possibile ottenere la classificazione delle parole ottenute, al fine di poter valutare quanto sia effettivamente possibile riconoscere una parola digitata dall'utente. Quindi, i casi di studio presentati successivamente, si concentreranno sui valori ricavati dai risultati presentati nella Sezione 4.5. A tal fine verranno considerate le seguenti percentuali: 40% e 60%, le quali rappresentano il *rate* di riconoscimento relativo al caso di studio *KBD* mostrato nelle Figure 4.6 e 4.7. Inoltre, verranno anche mostrati i casi in cui

si ottenga un riconoscimento pari all'80%, basato sull'ipotesi di un possibile miglioramento dei risultati ottenuti, e un riconoscimento pari al 100% posto come caso limite.

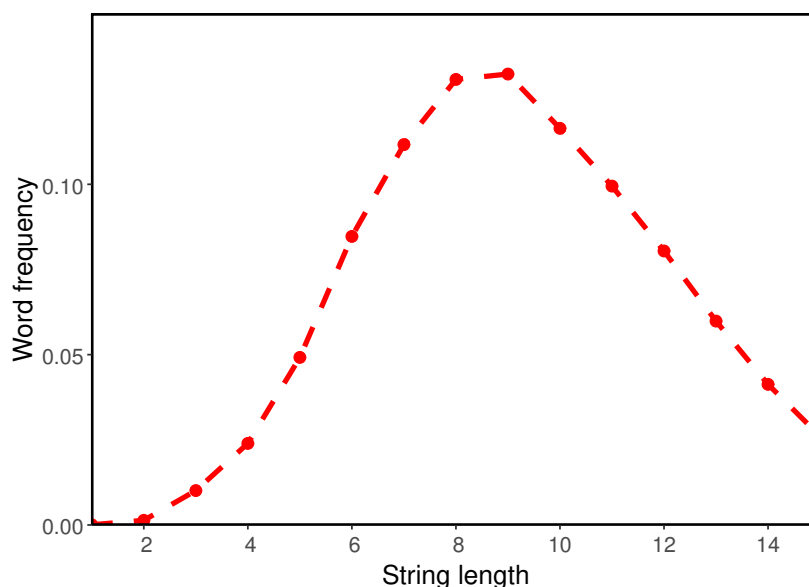
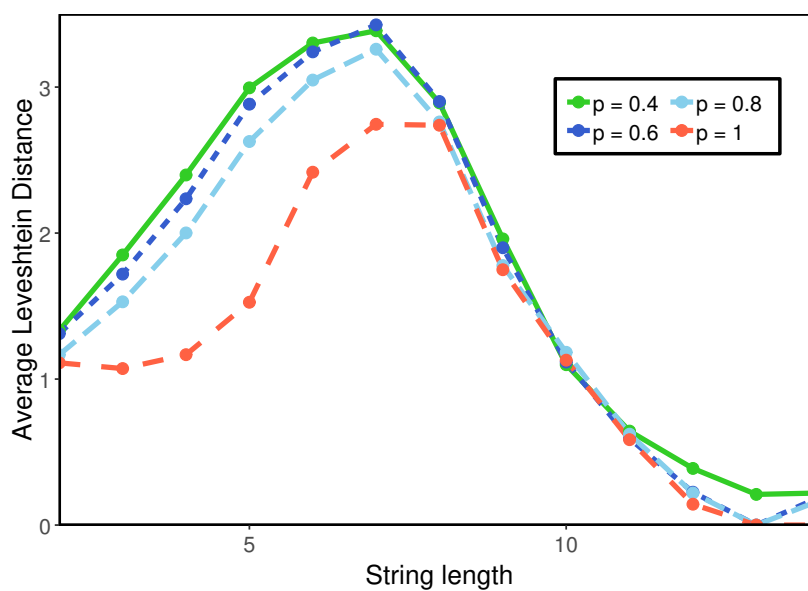
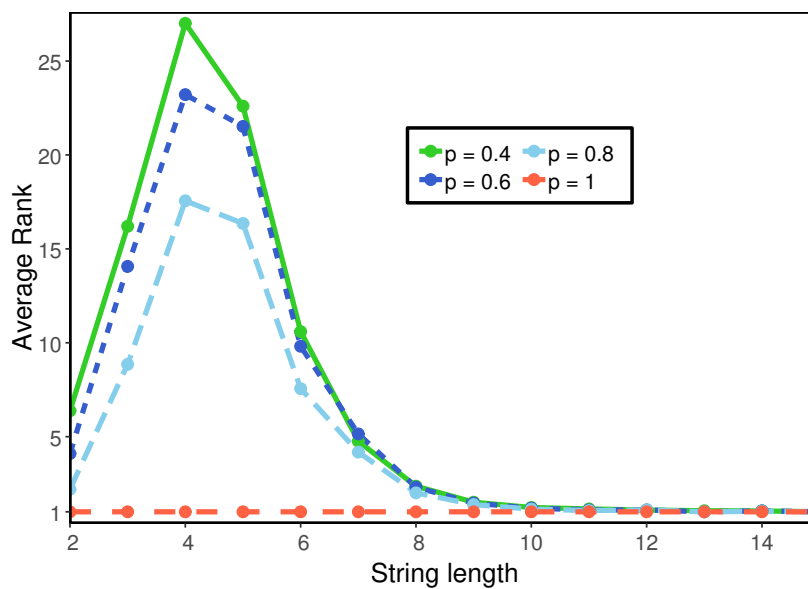


Figura 4.10: Frequenza delle parole in relazione alla lunghezza del dizionario considerato.

I grafici mostrati dalle Figure 4.10 4.11, 4.12 e 4.13 hanno tutti in comune i valori relativi all'asse X: tali valori rappresentano la lunghezza delle parole prelevate e successivamente studiate a partire da un dizionario inglese di oltre 470000 parole. Un ulteriore tratto in comune ai grafici considerati é il valore relativo alla probabilità. Le percentuali considerate sono pari a quelle descritte alla fine delle sezione precedente, cioè: 40%,60%,80% e 100%. Quest'ultimo valore viene considerato come caso limite, per il quale ci si aspetta di ottenere i migliori risultati di riconoscimento.

Dopo aver descritto i tratti in comune ai grafici considerati, é possibile adesso considerarli e analizzarli piú nel dettaglio.

Il grafico mostrato in Figura 4.10 non é il risultato di nessun calcolo relativo all'albero delle possibili parole digitate. Eppure, una sua analisi dettagliata risulta particolarmente utile, poiché mostra la frequenza delle parole presen-

Figura 4.11: *Distanza di Levenshtein Media.*Figura 4.12: *Media rank della parola cercata.*

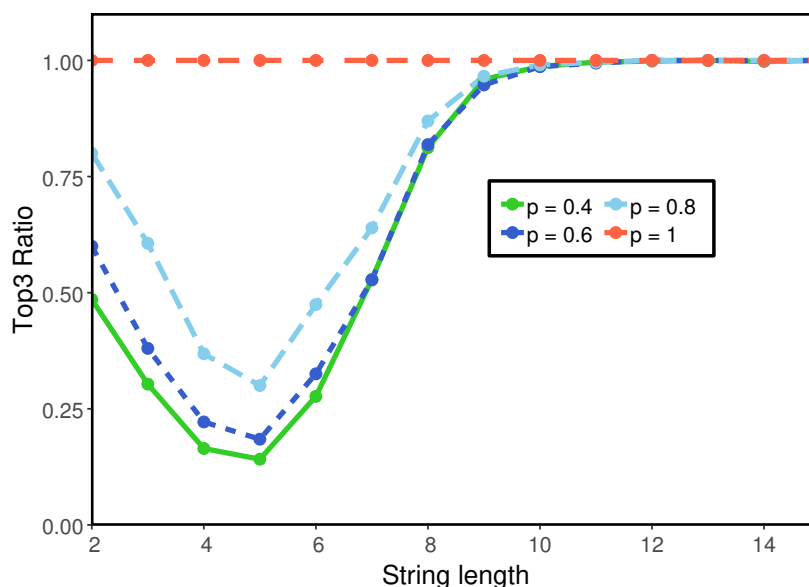


Figura 4.13: Accuratezza del riconoscimento.

ti nel dizionario considerato in relazione al numero dei caratteri da cui sono composte le parole stesse. Sulla base di tali considerazioni, risulta evidente come all'interno del dizionario vi sia una maggiore frequenza di parole composte da 8 o 9 caratteri. Il grafico mostra, inoltre, una frequenza minore in corrispondenza di parole che hanno una lunghezza inferiore o maggiore a tali valori. Quindi, potrebbe teoricamente risultare più difficile identificare parole da 8 o 9 caratteri, poiché potrebbero esistere più parole in corrispondenza dell'albero generato a partire dalla parola corretta considerata. Questa ipotesi teorica viene però smentita dai grafici relativi all'albero delle possibili parole testate a partire dal dizionario. Infatti, prendendo in considerazione il grafico in Figura 4.11, viene mostrato il rapporto tra la *distanza di Levenshtein* media ottenuta come risultato dei *test* effettuati sulle parole presenti all'interno del dizionario e sulla loro lunghezza. Si può notare, come ci si aspettava, che al crescere della probabilità associata ad ogni carattere digitato la *distanza di Levenshtein* diminuisce, mostrando un migliore riconoscimento delle parole prese in considerazione. È anche possibile notare un picco in corrispondenza di un valore pari a 7: tale picco mostra come per parole di lunghezza inferiore

a 5 e superiore a 8, sia possibile ottenere una distanza media relativamente bassa. Il primo risultato, ottenuto in corrispondenza di parole di lunghezza inferiore a 5, sarà, perciò, dovuto alla brevità delle parole considerate. Il secondo risultato, invece, evidenzia un migliore riconoscimento nel caso in cui si considerino parole di lunghezza superiore a 7. Tale considerazione è evidenziata dal fatto che si ottiene una distanza media leggermente inferiore in corrispondenza di parole di 8 caratteri rispetto ai risultati ottenuti in corrispondenza di parole di 5 caratteri. Inoltre, tali considerazioni evidenziano una maggiore accuratezza nella costruzione dell'albero delle possibilità nel caso in cui si considerino parole più lunghe rispetto alla costruzione dell'albero generato nel caso in cui si considerino parole di non oltre 5 caratteri. Quest'ultimo albero produrrà, inoltre, un maggior numero di foglie e quindi di parole possibili.

Il grafico mostrato in Figura 4.12 mostra, invece, il *rank* medio delle parole calcolate dall'albero delle possibilità in relazione alla loro lunghezza. Anche in questo caso, come si ci aspettava, i risultati ottenuti migliorano in proporzione all'aumento della probabilità considerata. Nel caso di tale grafico, è anche possibile notare come esso segua un andamento simile al grafico mostrato in Figura 4.10: l'unica differenza è rappresentata dal fatto che il grafico in questione registra un picco in corrispondenza di stringhe di lunghezza 4 piuttosto che in corrispondenza di stringhe di lunghezza 8. Tale considerazione rafforza ulteriormente la tesi secondo cui sia effettivamente più semplice riconoscere parole relativamente lunghe rispetto a parole formate da 5 o meno caratteri. Non a caso, il grafico evidenzia in particolare modo come, considerando parole di lunghezza inferiore a 5, queste abbiano un *rank* medio molto alto, che rende particolarmente difficile il loro riconoscimento. Tuttavia, tale valore di *rank* diminuisce nel caso in cui vengano considerate parole di lunghezza superiore o uguale a 6 caratteri, raggiungendo valori particolarmente interessanti per parole da 8 caratteri in poi. Tali risultati rafforzano ulteriormente l'idea secondo cui sia possibile riconoscere parole relativamente lunghe: infatti, al di sopra dei 10 caratteri, le parole

vengono addirittura identificate univocamente a prescindere dalla probabilità considerata. Un ulteriore grafico relativo al *rank* tramite cui vengono classificate le parole é quello mostrato in Figura 4.13. Tale grafico mostra la percentuale di volte in cui la parola corretta si ripresenta nella *Top3* delle parole ottenute dall'albero delle possibilità. É, inoltre, possibile notare come il grafico abbia un andamento simile ma, al tempo stesso, inverso rispetto a quello in Figura 4.11. Infine, anche in questo grafico é evidente come la presenza di probabilità maggiori permetta un migliore riconoscimento anche in corrispondenza di parole formate da pochi caratteri. Sulla base di queste ultime considerazioni, é possibile notare, anche nel contesto di questo grafico, la possibilità di riconoscere con ottimi risultati parole composte da 8 o piú caratteri. Tali riconoscimenti vengono ottenuti a prescindere dalla probabilità ad essi associata, raggiungendo un riconoscimento totale per parole da 10 caratteri in su.

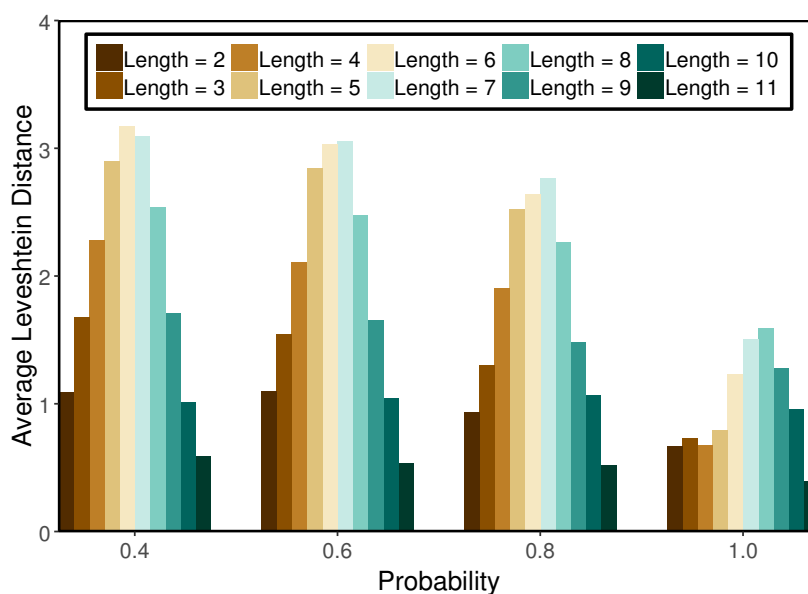


Figura 4.14: Distanza di Levenshtein media

Infine, un ulteriore grafico relativo ai risultati ottenuti dall'albero delle parole possibili, é rappresentato dal grafico a barre mostrato in Figura 4.14. Tale grafico differisce da quelli considerati precedentemente, perché ha come



valore dell'asse X le probabilità considerate e non la lunghezza della stringa. Infatti, la lunghezza delle parole verrà utilizzata, in questo caso, per differenziare i casi considerati all'interno del grafico stesso. Inoltre, anche nel contesto di questo grafico, è possibile notare come, in corrispondenza di probabilità maggiori, vengano prodotti risultati migliori. Tuttavia, quest'ultimo grafico mostra anche come sia possibile ottenere *una distanza di Levenshtein media* relativamente bassa nel caso in cui si ottenga una probabilità del 40% in corrispondenza di stringhe di lunghezza compresa tra 5 e 7. Il grafico conferma invece come, per i casi in cui la stringa avrà una lunghezza compresa tra 9 e 10, sia possibile ottenere una distanza di Levenshtein molto bassa, la quale rimane costante a prescindere dalla probabilità considerata. Tali considerazioni confermano ulteriormente il precedente risultato, secondo cui è più semplice riconoscere parole relativamente lunghe.



# Capitolo 5

## Conclusioni

In definitiva possiamo concludere che, nel contesto di questo elaborato, é stato possibile mostrare come, basandosi sui principi sviluppati nei diversi esempi presentati all'interno dello *Stato dell'arte*, si sia ottenuto un riconoscimento delle parole digitate da un utente. A tal fine sono stati sfruttati i dati relativi ai sensori di accelerometro e magnetometro, i quali hanno permesso di interpretare gli spostamenti effettuati dal dispositivo nello spazio.

A tal proposito, risulta tuttavia necessario ipotizzare che, l'utente in questione abbia effettuato i *touch events* sul proprio dispositivo utilizzando una sola man. In tal modo, é stato possibile eseguire dei *tilt* del dispositivo piú ampi in corrispondenza della digitazione di ogni carattere. Tale operazione ha reso possibile l'ottenimento di un migliore riconoscimento della sezione di schermo selezionata. Inoltre, sulla base di tali considerazioni, é stato mostrato come, in particolare, sia possibile riconoscere parole da 8 o piú caratteri con ottime percentuali di successo e parole piú brevi con una percentuale di successo pari circa all'80%. Tali operazioni, come già anticipato nell'Introduzione, potrebbero così mettere a rischio la privacy degli utenti dotati di uno *smartphone*.

## Lavori Futuri

Al fine di ottenere una maggiore precisione nel riconoscimento dei *touch events* effettuati da un'utente sul proprio dispositivo, sarebbe interessante prendere in considerazione lo studi di alcuni nuovi casi, i quali possano permettere di ampliare ulteriormente i risultati degli studi già affrontati all'interno di questo elaborato. Uno dei principali limiti del principali caso considerato all'interno di questo progetto é l'assenza del sensore giroscopio all'interno del dispositivo utilizzato per effettuare i *tests*. Infatti, i dati relativi alle misurazioni effettuate dal giroscopio consentirebbero di studiare anche la rotazione del dispositivo nello spazio: avere accesso a tali dati permetterebbe di ottenere un migliore riconoscimento del movimento dei *tilts* sul dispositivo considerato. Potrebbe anche rivelarsi interessante studiare la fase di transizione tra un *touch event* e l'altro, al fine di ottenere dei *patterns* piú precisi che permettano di descrivere il *touch event* appena avvenuto. Un ulteriore limite relativo al progetto attualmente sviluppato, é determinato dal fatto di aver potuto addestrare il modello di *machine-learning* solo sulla base dei dati ottenuti da un singolo dispositivo mobile. Infatti, potrebbe rivelarsi interessante basarsi sull'analisi di *database* contenenti i dati relativi a piú dispositivi, al fine di addestrare il modello di *machine-learning* con maggiore precisione. Un ultimo caso non preso in considerazione all'interno di questo elaborato, il quale potrebbe anche risultare triviale ai fini di un effettivo attacco da parte di un'applicazione malevola, é rappresentato dal caso in cui il modello di *machine-learning* venga addestrato non solo al riconoscimento dei *touch events* effettuati con una singola mano, ma anche al riconoscimento dei *touch events* con entrambe le mani o con una sola delle due, differenziando tra la destra e la sinistra. Tutto ciò renderebbe possibile, nella maggior parte dei casi, il riconoscimento dei caratteri digitati da un'utente, mettendone così a rischio la *privacy*.

# Bibliografia

- [1] Ling Bao and Stephen S. Intille. Activity recognition from userannotated acceleration data. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing*, pages 1 - 17, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
  
- [2] Luca Bedogni, Marco Di Felice, and Luciano Bononi. Context-aware Android applications through transportation mode detection techniques. *Wireless communications and mobile computing* 16.16(2016) 2523-2541.
  
- [3] Philippe Golle and Kurt Partridge. On the anonymity of home/work location pairs. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A. J. Bernheim Brush, and Yoshito Tobe, editors, *Pervasive Computing*, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
  
- [4] Y. Gu, Y. Yao, W. Liu, and J. Song. We know where you are: Home location identification in location-based social networks. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, Aug 2016.
  
- [5] Inhwon Hwang, Jungchan Cho, and Songhwai Oh. Privacy-aware communication for smartphones using vibration. *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2012.

- 
- [6] Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K. S. (2015). The internet of things for health care: a comprehensive survey. *IEEE Access*, 3, 678 – 708.
- [7] G. Li, L. Huang, and H. Xu. iwalk: Let your smartphone remember you. In 2017 4th International Conference on Information Science and Control Engineering (ICISCE), July 2017.
- [8] Mahmud, Jalal, Jeffrey Nichols, and Clemens Drews. "Home location identification of twitter users." *ACM Transactions on Intelligent Systems and Technology (TIST)* 5.3 (2014): 47.
- [9] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He. Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, ASIA CCS '16, New York, NY, USA, 2016. ACM.
- [10] Anindya Maiti, Murtuza Jadliwala, Jibo He, and Igor Bilogrevic. (smart)watch your taps: Side-channel keystroke inference attacks using smartwatches. In Proceedings of the 2015 ACM International Symposium on Wearable Computers, ISWC '15, New York, NY, USA, 2015. ACM.
- [11] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: Password inference using accelerometers on smartphones. In Proceedings of the Twelfth Workshop on Mobile Computing Systems ; Applications, HotMobile '12, New York, NY, USA, 2012. ACM.
- [12] Veljko Pejovic and Mirco Musolesi. Anticipatory Mobile Computing: A Survey of the State of the Art and Research Challenges. *ACM Computing Surveys (CSUR)*, 2015.
- [13] S. Sen, K. Grover, V. Subbaraju, and A. Misra. Inferring smartphone keypress via smartwatch inertial sensing. In 2017 IEEE International

- Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), March 2017.
- [14] Ahren Studer, Timothy Passaro, and Lujo Bauer. Don't bump, shake on it: The exploitation of a popular accelerometer-based smart phone exchange and its secure replacement. In Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC '11, New York, NY, USA, 2011. ACM.
- [15] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. 2017 IEEE European Symposium on Security and Privacy (EuroSP), 2017.
- [16] Q. Xia, F. Hong, Y. Feng, and Z. Guo. Motionhacker: Motion sensor based eavesdropping on handwriting via smartwatch. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), April 2018.
- [17] Tuo Yu, Haiming Jin, and Klara Nahrstedt. Writinghacker: Audio based eavesdropping of handwriting via mobile devices. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '16, New York, NY, USA, 2016.
- [18] Carpineti, C., Lomonaco, V., Bedogni, L., Di Felice, M., Bononi, L. (2018, March). Custom Dual Transportation Mode Detection by Smartphone Devices Exploiting Sensor Diversity. In 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). IEEE.
- [19] Luca Bedogni, and Marco Levorato. "Rising User Privacy Against Predictive Context Awareness Through Adversarial Information Injection." 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018.

