Home Zone Analyzer

1st Mirco Chiarini
Dipartimento di Informatica
Università di Bologna
Bologna, Italia
mirco.chiarini@studio.unibo.it

2nd Emanuele Corsi
Dipartimento di Informatica
Università di Bologna
Bologna, Italia
emanuele.corsi3@studio.unibo.it

3rd Antonio Sisinni
Dipartimento di Informatica
Università di Bologna
Bologna, Italia
antonio.sisinni@studio.unibo.it

***Abstract*— Trovare la casa ideale può essere un compito complesso e stressante, specialmente in una città ricca di opportunità come Bologna. Per rispondere a questa esigenza, proponiamo lo sviluppo di Home Zone Analyzer, una piattaforma software avanzata per l'analisi delle location, focalizzata sull'ottimizzazione del processo di acquisto immobiliare basato sulle preferenze personali degli utenti. La piattaforma, integrando tecnologie di analisi spaziale, offre suggerimenti personalizzati sull'acquisto di immobili. Home Zone Analyzer è composta da un back-end che gestisce i dati spaziali tramite POSTGRES/POSTGIS e un front-end sviluppato con OpenLayers, permettendo un'interazione intuitiva con le mappe. La piattaforma viene sviluppata utilizzando container Docker orchestrati con Kubernetes, garantendo scalabilità e facilità di gestione. In sintesi, Home Zone Analyzer rappresenta una soluzione innovativa per l'analisi spaziale e l'ottimizzazione delle decisioni di acquisto immobiliare, integrando tecnologie di gestione dei dati e interazione utente.**

Keywords— location analytics, spatial data processing, PostgreSQL, PostGIS, OpenLayers, Docker, Kubernetes, point of interest, geofence, Moran's I.

I. INTRODUZIONE

La crescente digitalizzazione e l'espansione dei dati spaziali hanno aperto nuove opportunità per lo sviluppo di applicazioni innovative nel settore della gestione immobiliare. In questo contesto, si inserisce il progetto di sviluppo di una piattaforma software avanzata dedicata alla gestione e al processamento di dati spaziali, con un focus specifico sulla città di Bologna. La piattaforma proposta ha l'obiettivo di supportare gli utenti nella ricerca e selezione della posizione ideale per l'acquisto di un immobile, basandosi su preferenze espresse dagli utenti riguardo a diverse tipologie di Punti di Interesse (PoI) presenti nel loro vicinato.

A. Obiettivi del progetto

Il progetto ha come principale obiettivo la creazione di una soluzione software composta da due componenti principali: un back-end robusto e un front-end interattivo. Il sistema deve essere in grado di:

1. **Sottoporre un Questionario all'Utente**: La piattaforma fornisce un questionario dettagliato che consente di raccogliere le preferenze dell'utente riguardo alla disponibilità e densità di PoI nel vicinato dell'immobile. Gli utenti esprimono un gradimento su una scala da 0 a 5 per almeno venti diverse tipologie di PoI, come aree verdi, parcheggi e fermate bus, palestre, scuole, etc.
2. **Gestire Marker e Geofence su Mappa**: L'utente può selezionare aree su una mappa per ricercare immobili di interesse e definire geofence poligonali per individuare aree di interesse. Entrambe le operazioni vengono effettuate interattivamente attraverso la mappa.
3. **Visualizzare Dati dei PoI e Applicare Filtri**: La piattaforma consente di visualizzare i PoI su mappa con la possibilità di filtrarli in base alla loro tipologia, facilitando così l'esplorazione dei punti d'interesse disponibili nella zona.
4. **Calcolare e Visualizzare un Ranking degli Immobili e delle Aree**: Gli immobili e le aree candidate vengono visualizzati con marker colorati che riflettono un punteggio basato su una funzione di ranking. Questo punteggio rappresenta quanto una località soddisfa le preferenze dell'utente, considerando i PoI presenti nel vicinato.
5. **Modificare i Parametri di Vicinato e Calcolare l'Indice di Moran's I**: La piattaforma offre la possibilità di configurare i parametri del vicinato e calcolare l'indice di Moran per analizzare la relazione spaziale tra il prezzo medio al metro quadro degli immobili e i dati dei PoI.
6. **Suggerire Posizioni Ottimali per l'Acquisto di Immobili**: Basandosi sui dati raccolti e sulle preferenze dell'utente, il sistema suggerisce una o più posizioni ottimali per l'acquisto di una casa a Bologna.
7. **Un meccanismo per la valutazione della vicinanza dei PoI**: Basandosi sul tempo di raggiungimento, anziché sulla sola distanza a linea d'aria.
8. **Informazioni Personali:** Una sezione dedicata per l'inserimento di informazioni personali relative ai luoghi che l'utente visita giornalmente, che permette di selezionare punti d'interesse personali e ottenere percorsi ottimali per raggiungerli.
9. **Aggiornare i PoI:** La possibilità di modificare i PoI per visualizzare come le modifiche influenzano la ricerca dell'immobile.
10. **Implementare un Meccanismo di Predizione dei Prezzi**: La piattaforma include un sistema di predizione che permetta di visualizzare l'andamento futuro dei prezzi al metro quadro delle diverse zone di Bologna. Questo strumento aiuterà

gli utenti a prendere decisioni informate sull'acquisto di immobili.

II. Metodologia

A. Architettura del Sistema

A1. Backend

Il backend della nostra applicazione è gestito con Express, utilizzando metodi GET e POST per comunicare con il database, eseguire query e gestire l'invio e la ricezione di dati dal frontend. In aggiunta, abbiamo una cartella contenente file Python che implementano il calcolo dell'indice di Moran, il quale misura la correlazione spaziale dei prezzi al metro quadro degli immobili in base alla zona di Bologna. Inoltre, questi file Python si occupano della predizione dei prezzi futuri al metro quadro nelle varie zone utilizzando un modello di random forest.

Questa cartella include:

- I file di esecuzione dei modelli.
- Il modello di predizione generato.
- I file JSON con i dati di input per la random forest, prelevati dal sito Idealista, comprendenti i prezzi annuali degli immobili al metro quadro dal 2016 al 2023/2024 per ogni zona di Bologna.
- I file JSON con i dati generati dal calcolo dell'indice di Moran.

La random forest viene utilizzata per predire i futuri prezzi al metro quadro degli immobili per una serie di anni. Parallelamente, i file che calcolano l'indice di autocorrelazione spaziale includono una sezione che calcola anche la correlazione "locale". Questo significa che viene analizzata la correlazione di prezzo tra una zona di Bologna e le sue zone "neighbour", evidenziando relazioni come HH (High-High), ovvero zone con un prezzo elevato circondate da altre zone con prezzi altrettanto elevati.

In sintesi, il backend di questa applicazione offre una robusta integrazione tra il server Express e gli script Python, permettendo sia la gestione delle richieste del frontend sia l'analisi e la predizione dei prezzi immobiliari basata su dati storici e spaziali.

A2. Frontend

Per il frontend della nostra applicazione, abbiamo utilizzato una combinazione di tecnologie e librerie per fornire un'interfaccia utente interattiva e funzionale. Il frontend è sviluppato con:

- **React:** Utilizzato come framework principale per costruire l'interfaccia utente. React permette la creazione di componenti modulari e riutilizzabili, facilitando la gestione dello stato e il rendering dinamico dei contenuti.
- **Shadcn:** Utilizzato per i componenti dell'interfaccia utente. Shadcn fornisce una serie di componenti predefiniti che migliorano l'aspetto visivo e la coerenza del design dell'applicazione.
- **Lucide React:** Utilizzato per le icone. Lucide React fornisce un set di icone vettoriali scalabili che possono essere facilmente integrate nei componenti di React, migliorando l'usabilità e l'intuitività dell'interfaccia utente.
- **MUI X Charts:** Impiegata per estendere velocemente i componenti di grafici per la visualizzazione dei dati. Inoltre, come gli altri componenti MUI X, i grafici sono pronti per la produzione, integrandosi perfettamente e offrendo un elevato livello di personalizzazione.
- **Vite:** Utilizzato come build tool per il progetto. Vite offre una configurazione rapida e leggera per lo sviluppo frontend, con una gestione efficiente dei moduli e un'ottimizzazione delle performance.
- **OpenLayer:** Utilizzato per la gestione delle mappe. OpenLayer consente l'integrazione e la visualizzazione di mappe interattive, supportando funzionalità avanzate come il disegno di geofence e la visualizzazione di punti di interesse.

Il frontend è strutturato in varie pagine e componenti che gestiscono le diverse sezioni dell'applicazione. Questi componenti comunicano con il backend tramite chiamate fetch (GET o POST), ad esempio le usiamo per inviare i dati del questionario compilato dagli utenti e interagire con il database o per inviare geofence disegnati dall'utente e recuperare i punti di interesse all'interno di tali aree.

Abbiamo inoltre integrato un servizio di API per il calcolo delle isocrone e del percorso ottimale per raggiungere i luoghi visitati giornalmente dagli utenti. Questo servizio, fornito da Open Route Service, consente di:

- Calcolare le isocrone, ovvero le zone sulla mappa raggiungibili in un determinato periodo di tempo utilizzando un mezzo di trasporto specifico.
- Determinare il percorso ottimale giornaliero, tenendo conto di eventuali waypoint intermedi. Questa funzionalità simula scenari in cui l'utente potrebbe voler fare determinate soste durante un tragitto classico che effettua nella sua quotidianità.

In aggiunta, è stato utilizzato anche Geoapify per il geocoding, cioè per cercare un indirizzo e ottenere la

posizione corrispondente. Questa funzionalità è stata impiegata per scegliere il luogo dal quale calcolare un'area con un determinato raggio, all'interno della quale visualizzare le case disponibili.

In sintesi, il frontend dell'applicazione utilizza una combinazione di tecnologie avanzate per fornire un'interfaccia utente dinamica e interattiva, che comunica in modo efficiente con il backend e supporta funzionalità avanzate come la gestione delle mappe e il calcolo dei percorsi ottimali.

A3. Container e Orchestrazione

L'applicazione è gestita e containerizzata utilizzando Docker e orchestrata con Kubernetes per garantire una distribuzione scalabile e robusta. Di seguito, viene fornita una descrizione dettagliata delle tecnologie e dei processi coinvolti.

A3.1. Docker

Dockerfile Backend: Un Dockerfile specifico per il backend che include l'avvio dell'index e dei file python, oltre all'installazione delle librerie necessarie. Utilizziamo un'immagine combinata Python-Node presa da Docker Hub per supportare sia l'ambiente Node.js che quello Python, necessari per il nostro backend.

Dockerfile Frontend: Un Dockerfile per il frontend che utilizza un'immagine Node presa da Docker Hub. Questo file si occupa di caricare l'applicazione front-end e avviarla.

Docker Compose: Utilizziamo un file docker-compose.yml per definire e gestire i vari servizi dell'applicazione. Questo include:

- **Database:** Un servizio che utilizza l'immagine PostGIS, con specifiche per le credenziali, la porta e il volume per la persistenza dei dati.
- **pgAdmin**: Un servizio per la gestione del database tramite interfaccia grafica.
- **Backend e Frontend:** Riferimenti ai container del backend e frontend definiti nei rispettivi Dockerfile.

A3.2. Kubernetes

File YAML: Definiamo vari file YAML per descrivere la configurazione e il deployment dei componenti dell'applicazione:

Backend e Frontend: File YAML che specificano la configurazione e il deployment dei pod per il backend e il frontend.

pgAdmin e Postgres: File YAML per il deployment di pgAdmin e del database Postgres, inclusi i dettagli del Persistent Volume per garantire la persistenza dei dati.

Docker Hub Repository: Dopo aver creato le immagini Docker per il frontend e il backend, le carichiamo nelle nostre repository personali su Docker Hub. Questo permette una facile distribuzione delle immagini durante il deployment su Kubernetes.

Deployment con kubectl: Utilizziamo il comando kubectl apply -f "file.yaml" per eseguire i vari file YAML. Questo comando applica la configurazione specificata nei file YAML e gestisce il ciclo di vita dei container.

B. Raccolta e gestione dei Dati

Per ottenere i dati spaziali necessari al progetto, sono state utilizzate le piattaforme OpenData e DatiOpen. Queste piattaforme sono state scelte per la loro capacità di fornire dati spaziali accurati e aggiornati. I dati sono stati scaricati in formato SHP, un formato comune per dati geografici che permette una facile manipolazione e importazione.

Una volta scaricati i file SHP, è stato utilizzato il comando shp2pgsql per convertire i file SHP in formato SQL compatibile con il database POSTGRES. Questa conversione ha permesso di importare agevolmente i dati spaziali nel database POSTGRES/POSTGIS, facilitando così la loro gestione e storicizzazione. Questo processo di conversione e importazione è stato applicato alla quasi totalità dei dataset utilizzati nella piattaforma.

Tuttavia, per i dati utilizzati nelle funzionalità di predizione dei prezzi e per il calcolo dell'indice di Moran's I, è stato necessario un approccio differente. Questi dati sono stati reperiti da Idealista, un sito web specializzato nel settore immobiliare, che fornisce dataset storici sui valori degli immobili. Questi dati sono stati elaborati separatamente e integrati nel database POSTGRES per essere utilizzati nelle analisi e nei calcoli predittivi della piattaforma.

L'intero processo di raccolta e gestione dei dati è stato cruciale per garantire che la piattaforma potesse fornire suggerimenti accurati e utili agli utenti, basati su informazioni spaziali e storiche di alta qualità.

C. Implementazione delle Funzionalità

C1. PoI

Abbiamo implementato diverse funzionalità all'interno della nostra web app per migliorare l'esperienza utente e fornire suggerimenti accurati sugli immobili. Le preferenze dell'utente sono gestite tramite un questionario in cui l'utente può esprimere un gradimento da 0 a 5 per diversi Punti di Interesse (PoI). I dati raccolti dal questionario vengono salvati e utilizzati per calcolare un punteggio complessivo basato sul numero di PoI nelle vicinanze di ogni immobile nelle aree selezionate.

Il punteggio di ogni PoI contribuisce a uno score complessivo che determina il ranking degli immobili. Gli immobili sono visualizzati su una mappa con una colorazione che va dal verde (posizione molto consigliata) al rosso (posizione meno consigliata), aiutando l'utente a effettuare una scelta informata. Inoltre, l'utente può

visualizzare e filtrare i PoI sulla mappa, aggiungendo o rimuovendo PoI in base alle proprie preferenze, permettendo così una personalizzazione della ricerca.

Inoltre, tramite lo score è stato possibile generare e ottenere la lista delle aree consigliate. Questa lista è basata su una funzione di ranking espressa tramite una colorazione che va dal verde (molto consigliata) al rosso (meno consigliata).

C2. Predizione dei Prezzi

Il meccanismo di predizione è stato implementato utilizzando il modello Random Forest. Per l'addestramento e il testing del modello, è stato utilizzato un dataset contenente informazioni sui prezzi medi al metro quadro negli ultimi 10 anni, reperito da Idealista. Il dataset è stato diviso in due parti: 80% per il training e 20% per il testing. Dopo l'addestramento, il modello è stato utilizzato per predire i prezzi futuri degli immobili. L'intero processo di predizione è stato implementato utilizzando librerie Python.

C3. Selezione aree

Abbiamo implementato tre diverse tipologie di selezione delle aree:

1. Visualizzazione dei quartieri con possibilità di interazione tramite click.

2. Disegno a mano libera di una o più geometrie sulla mappa.

3. Ricerca per indirizzo e visualizzazione di una geofence rotonda con possibilità di modificare il raggio di ricerca.

C4. Clustering

Abbiamo utilizzato il clustering K-means per individuare i centri ottimali dei cluster. Il centroide di ciascun cluster è considerato come posizione ottimale consigliata basata sulle utilizzato preferenze dell'utente espresse tramite i PoI.

C5. Tempo Raggiungimento dei PoI

Il calcolo del tempo di raggiungimento dei PoI è stato implementato utilizzando l'API di Open Route Service. L'API permette di selezionare il mezzo di trasporto e calcola il raggio delle zone raggiungibili in un determinato numero di minuti, migliorando così l'accuratezza delle raccomandazioni basate sulla vicinanza temporale dei PoI.

C6. Indice di Moran's

L'indice di Moran's I è stato calcolato utilizzando Python per analizzare la correlazione spaziale tra il prezzo medio degli immobili e la zona di appartenenza. Questo indice ha permesso di ottenere un'analisi approfondita della distribuzione spaziale dei prezzi degli immobili in relazione ai PoI presenti nelle vicinanze.

III. Discussione e Risultati

A. Random Forest

Per valutare le prestazioni del modello abbiamo preso in considerazione i seguenti parametri:

- MAE (Mean Absolute Error) rappresenta l'errore medio assoluto. È la media degli errori assoluti tra i valori previsti e i valori reali.

  Training MAE: 20.901 significa che in media, l'errore assoluto del modello sui dati di addestramento è di circa 20.901 unità (euro/m²).

  Test MAE: 59.567 indica che in media, l'errore assoluto del modello sui dati di test è di circa 59.567 unità(euro/m²). Questo valore è più alto rispetto al Training MAE, suggerendo che il modello potrebbe non generalizzare bene sui dati non visti.

- RMSE (Root Mean Squared Error) rappresenta la radice quadrata dell'errore quadratico medio. Misura la deviazione standard degli errori di previsione.

  Training RMSE: 31.767 indica la media quadratica degli errori sui dati di addestramento. Un valore più alto rispetto alla MAE, indicando la presenza di alcuni errori più grandi.

  Test RMSE: 116.128 mostra la media quadratica degli errori sui dati di test. Anche questo valore è notevolmente più alto rispetto al Training RMSE, rafforzando l'idea che il modello potrebbe avere difficoltà a generalizzare.

Per cui, Il Test MAE è significativamente più alto del Training MAE, suggerendo che il modello potrebbe essere overfit, cioè si adatta molto bene ai dati di addestramento ma non riesce a generalizzare sui dati non visti. Un pattern simile si osserva con l'RMSE, dove il valore di test è molto più alto del valore di addestramento. L'RMSE più alto sottolinea ulteriormente la presenza di errori più grandi nei dati di test.

B. Indice di Moran

L'indice di Moran è una misura di autocorrelazione spaziale che quantifica il grado di similarità tra valori in una regione geografica. Questo indice varia tipicamente tra -1 e +1, indicando rispettivamente una forte autocorrelazione negativa o positiva. Nel nostro caso, l'indice di Moran di 0.36 suggerisce una moderata autocorrelazione spaziale positiva. Ciò indica che, in media, le aree vicine tendono ad avere valori simili, sebbene non in maniera estremamente forte, implicando che i prezzi degli immobili mostrano tendenze simili nelle aree adiacenti.

Tuttavia, l'indice di Moran da solo potrebbe non essere sufficiente per stabilire la significatività statistica di questa autocorrelazione. Per questo motivo, è stato calcolato anche il p-value. Il valore ottenuto è molto basso, circa 0.009, il

che ci consente di rifiutare l'ipotesi nulla. Questo significa che esiste una forte evidenza che l'autocorrelazione spaziale osservata nei dati non sia il risultato del caso.

Nel calcolo dell'indice di Moran "locale", abbiamo adottato un valore comune per il p-value, ovvero 0.05, che corrisponde a un livello di significatività del 5%. Questo indica che c'è solo una probabilità del 5% di concludere erroneamente che esista un pattern spaziale quando in realtà non c'è. Questo parametro fornisce una buona sicurezza nei risultati, evidenziando che solo alcune zone presentano una correlazione spaziale significativa rispetto alle zone circostanti.

In particolare, le zone centrali di Bologna sono ben classificate come "HH", indicando che il prezzo al metro quadro degli immobili è elevato in queste aree e continua ad esserlo nelle zone adiacenti.

C. Limiti e Sfide

Inizialmente, avendo molte tipologie di punti di interesse (POI) e, quindi, molte tabelle con grandi quantità di dati, abbiamo riscontrato significative limitazioni nelle prestazioni delle query. Abbiamo scaricato dataset completi degli edifici e dei POI, generando tabelle contenenti migliaia di righe. Ciò ha comportato tempi di esecuzione prolungati per le query quando cercavamo di assegnare i punteggi alle case o alle zone, poiché effettuavamo molti join tra tabelle. Per migliorare le prestazioni, abbiamo adottato due strategie principali:

- **Riduzione della Dimensione delle Tabelle:** Abbiamo ridotto la dimensione delle tabelle filtrando i dati non necessari e mantenendo solo quelli rilevanti per l'analisi. Questo ha aiutato a non degradare troppo le prestazioni dell'applicazione durante l'utilizzo, riducendo il carico computazionale e migliorando i tempi di risposta.

- **Creazione di Viste nei Database:** Abbiamo creato delle viste apposite per salvare i dati che avremmo utilizzato frequentemente. Le viste sono essenzialmente query predefinite memorizzate nel database, che permettono di semplificare l'accesso ai dati complessi e di ottimizzare le prestazioni delle query.

Un'altra sfida che abbiamo affrontato è stata la limitazione della quantità di dati che potevamo inviare nel payload di una richiesta POST. Nello specifico, nell'applicazione è possibile selezionare uno o più quartieri per visualizzare gli edifici in quell'area. Per fare questo, effettuiamo una richiesta POST dal frontend al backend inviando la geometria dei quartieri selezionati. Il problema è sorto con quartieri particolarmente complessi dal punto di vista geometrico, poiché i dati da inviare eccedevano i limiti del payload di Express. Per risolvere questo problema, abbiamo aumentato il limite del payload delle richieste POST fino a 50 MB, adottando un approccio prudente. Il limite di default di Express era troppo restrittivo, impedendoci di inviare anche solo 100 KB di dati. Aumentando questo limite, siamo stati in grado di gestire con successo la complessità geometrica dei quartieri selezionati, garantendo che l'applicazione potesse funzionare senza interruzioni.

IV. CONCLUSIONI

L'implementazione di Home Zone Analyzer ha portato alla realizzazione di una piattaforma innovativa che facilita l'acquisto immobiliare attraverso l'analisi spaziale e la personalizzazione delle preferenze degli utenti. Home Zone Analyzer combina un backend robusto con un frontend interattivo per offrire un'esperienza utente avanzata nella ricerca di immobili. L'utilizzo di PostgreSQL/PostGIS per la gestione dei dati spaziali e di OpenLayers per l'interazione con le mappe ha permesso di creare una piattaforma user-friendly e altamente funzionale. Home Zone Analyzer rappresenta un importante passo avanti nell'applicazione delle tecnologie di analisi spaziale nel settore immobiliare. La piattaforma può diventare uno strumento prezioso per agenti immobiliari e acquirenti, fornendo un supporto decisionale basato su dati spaziali e preferenze personalizzate.

A. Lavori Futuri

Un lavoro futuro potrebbe essere quello di migliorare il modello di predizione del prezzo degli immobili. Attualmente, il modello potrebbe soffrire di overfitting, ovvero si adatta troppo bene ai dati di addestramento e non riesce a generalizzare adeguatamente ai dati di test. Per affrontare questo problema, si potrebbe aumentare il volume dei dati a disposizione e/o esplorare e includere nuove caratteristiche che potrebbero influenzare il prezzo degli immobili, come dati socio-economici, trend storici dei prezzi, e variabili climatiche.

Oltre alla webapp esistente, si potrebbe anche sviluppare un'applicazione per smartphone. Questa app permetterà agli utenti di accedere alle funzionalità principali dell'applicazione web in mobilità, migliorando l'accessibilità e l'usabilità del servizio.

Per mantenere l'accuratezza e la rilevanza dei dati, si potrebbe sviluppare un sistema di aggiornamento continuo per il dataset degli edifici e dei punti di interesse (POI), magari collegandosi a fonti di dati esterne che forniscono aggiornamenti regolari.

Un'altra funzionalità potrebbe essere l'introduzione di una sezione dedicata ai proprietari di immobili che desiderano mettere in vendita le loro proprietà. Lo scopo è quello di permettere ai proprietari di inserire dettagli sui loro immobili, inclusi descrizioni, foto, e prezzi richiesti, andando poi ad aggiornare mappa e database con i nuovi edifici registrati dai proprietari, rendendoli visibili agli acquirenti potenziali.

REFERENCES

[1] PostgreSQL documentation: https://www.postgresql.org/docs/
[2] PostGIS documentation: https://postgis.net/documentation/
[3] OpenLayer: https://openlayers.org/
[4] DockerHub: https://hub.docker.com/
[5] Express documentation: https://expressjs.com/
[6] React documentation: https://react.dev/
[7] Shadcn documentation: https://ui.shadcn.com/

[8] Lucide react documentation: https://lucide.dev/guide/packages/lucide-react
[9] Vite documentation: https://vitejs.dev/
[10] Open route api: https://api.openrouteservice.org/
[11] Geoapify: https://www.geoapify.com/
[12] MUI X Charts: https://mui.com/x/react-charts/

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published