Analisi del comportamento degli utenti attraverso lo studio delle recensioni su TripAdvisor

1. **Introduzione**

Con l’avvento di Internet, i ristoratori hanno iniziato a pubblicizzare la propria attività tramite siti web e social network. Questa tendenza è risultata utile per due motivazioni:

* Maggiore visibilità per le attività di ristorazione, con conseguente aumento della clientela.
* Ricezione di feedback riguardanti il servizio fornito e quindi di effettuare modifiche volte a migliorare le loro offerte.

Le recensioni sono quindi diventate fondamentali per i ristoratori, in quanto sono uno degli strumenti utilizzati dai clienti per selezionare i ristoranti che più rispondono alle proprie esigenze e gusti. Tramite le recensioni è possibile intuire più precisamente quali servizi vengono maggiormente apprezzati e quali invece dovrebbero essere migliorati secondo l’utenza, ad esempio un cliente potrebbe apprezzare la cucina offerta da un ristorante, ma potrebbe esprimersi negativamente riguardo il personale.

Una delle piattaforme web più utilizzate nel campo delle recensioni online di ristoranti è TripAdvisor.

1. **TripAdvisor**

Tripadvisor, Inc. è un'azienda statunitense di viaggi e ristoranti che pubblica recensioni di alberghi, bed and breakfast e ristoranti, prenotazioni di alloggi e altri contenuti relativi ai viaggi. Include anche forum di viaggi interattivi ed è diffusa in tutto il mondo.

Il sito vanta un largo bacino di utenza tra cui: ristoratori e albergatori, i quali pubblicizzano le loro attività, e clienti, i quali utilizzano il sito sia come strumento di selezione, sia come intermediario per le loro prenotazioni oppure per lasciare recensioni.

Per la piattaforma cruciale è l’attività di recensione, infatti ogni anno vengono assegnati premi alle attività meglio recensite che ne attestano la qualità. Questi premi sono molto ambiti dalle attività poiché permettono di ottenere una maggiore visibilità al pubblico.

* 1. **Recensioni**

Le recensioni sono lo strumento con cui un cliente previa consumazione nell’attività di riferimento propone una critica nei confronti del locale. Qualsiasi persona dotata di un account registrato sulla piattaforma può pubblicare una propria recensione, indicando solitamente un titolo a scopo esplicativo, un corpo della recensione ed una votazione che va da 1 a 5 stelline, più è alto il valore delle stelline più è stata positiva l’esperienza del cliente.

Più recensioni positive ottiene un locale, più questo acquisisce visibilità all’interno della piattaforma, tale meccanismo è molto simile alla presentazione delle pagine sui browser, i siti più ricercati sono tra le prime pagine, mentre quelle meno cercate sono “nascoste” nelle pagine successive.

Una recensione può riguardare diversi aspetti del locale recensito, ad esempio una recensione per un ristorante può parlare di cibo, location, costi e servizio offerto. Tali informazioni risultano indispensabili nello studio affrontato nell’elaborato proposto.

1. **Obiettivo del lavoro svolto**

Nel seguente elaborato ci siamo posto l’obiettivo di studiare ed analizzare il comportamento dei recensori all’interno della piattaforma, in particolare per le recensioni delle attività di ristorazione.

I ristoranti in questo caso di studio sono stati selezionati in base alla loro regione di appartenenza, principalmente abbiamo raccolto i ristoranti più famosi dei capoluoghi delle regioni italiane, più specificamente sono stati selezionati approssimativamente lo stesso numero di ristoranti per il Nord, Centro e Sud dell’Italia.

I capoluoghi selezionati sono (Da Nord a Sud):

* Ancona
* Bologna
* Milano
* Trieste
* Genova
* Venezia
* Firenze
* L’Aquila
* Roma
* Perugia
* Potenza
* Napoli
* Palermo
* Cagliari
* Catanzaro

Per ciascuna delle città selezionate, sono stati considerati i primi 13 ristoranti che comparivano inserendo la città come input di ricerca nel sito. Per ciascuno dei 13 ristoranti sono state considerate all’incirca 15 recensioni, per un totale di 2925 recensioni. I valori elencati hanno inoltre rappresentato il workload dello studio condotto.

Lo scopo del lavoro svolto è quello di individuare attraverso l’analisi delle recensioni, la quantità di recensioni che sono positive o comunque costruttive, la quantità di recensioni negative e quelle invece che si frappongono tra le due o più semplicemente quelle neutrali.

Questo tipo di analisi oltre ad essere fatto sul totale di tutte le recensioni per tutte le città selezionate, è stata eseguita anche su una suddivisione per zone della penisola, ovvero nord, sud e centro.

Una simile analisi, ma più capillare è stata fatta sugli aspetti delle recensioni, cioè fra tutte le recensioni rilevate, ne abbiamo estrapolato il contesto e ne abbiamo analizzato il sentimento, per comprendere se quest’ultimo è stato positivo, negativo o neutrale. Gli aspetti analizzati riguardano essenzialmente le caratteristiche principali di un’attività di ristorazione quali il cibo, la location, il costo e il servizio offerto. Tale analisi è utile quindi a capire più precisamente quali aspetti del ristorante sono stati apprezzati maggiormente e quali invece sono stati criticati.

* 1. **Tecnologie e metodologie utilizzate**

L’estrazione dei dati è stata costruita attraverso l’impiego di un tool di scraping pubblicato da Maximilian Copelli sul sito [www.apify.com](http://www.apify.com). Il tool consente di prelevare le recensioni e i ristoranti in base ad un input predefinito, in particolare la città, restituendo le informazioni richieste in un file JSON. Le informazioni contenute all’interno del file restituito comprendono:

* Nome del ristorante
* Città in cui è ubicato il ristorante
* Il tipo di cucina del ristorante, ad esempio: sushi, orientale, pizzeria, etc…
* Le recensioni

Per la lettura dei dati e per le elaborazioni core del progetto è stato utilizzato Java 8, che ci ha concesso di prelevare i dati dai file ed organizzare i risultati ottenuti dalla fase di sentiment analysis in file di output.

La sentiment analysis e l’opinion mining delle recensioni sono stati realizzati con i cognitive services offerti da Microsoft Azure. In particolare l’operazione di sentiment analysis ci ha permesso di valutare il sentiment di una particolare recensione, mentre l’opinion mining ci ha permesso di estrapolare le parole chiave all’interno della recensione e di fornire anche in questo caso un sentiment correlato.

Per utilizzare i cognitive services si necessita di un account iscritto alla piattaforma di Azure e di effettuare una subscription a tali servizi. La subscription fornisce delle chiavi di accesso collegate all’account che consentono di autenticare un client che verrà utilizzato per richiamare i metodi offerti dal servizio. I metodi sono chiamati attraverso semplici istruzioni Java e restituiscono risposte in formato JSON.

L’autenticazione è stata effettuata come segue:

**public** **static** **void** inizialize(String key, String endpoint) {

*client* = TextAnalyticsSamples.*authenticateClient*(key, endpoint);

*client* = TextAnalyticsSamples.*authenticateClient*(key, endpoint);

}

Dove key è la chiave di sottoscrizione, ed endpoint è l’indirizzo della risorsa in cloud.

Le operazioni di sentiment analysis, invece sono state effettuate come segue:

JSONObject sentimentInfo = **new** JSONObject();

sentimentInfo = TextAnalyticsSamples.*sentimentAnalysisWithOpinionMining*(*client*,textOfReview);

* 1. **Limitazioni sullo studio**

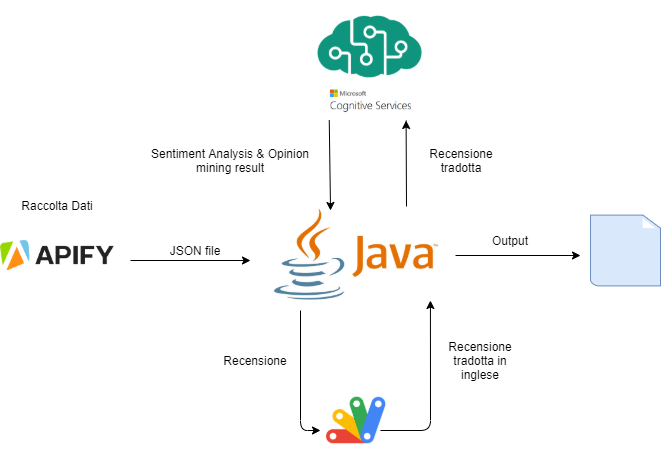
Le limitazioni riscontrate riguardano soprattutto l’impiego di tecnologie, ad esempio per Apify non è stato possibile prelevare le informazioni delle recensioni per alcune città, in particolare per la città di Bari, l’output non era veritiero, infatti i ristoranti rilevati erano ubicati in Argentina.

Non è stato possibile, inoltre prelevare più di 13 recensioni per ristorante, poiché i cognitive services di Azure fornivano un numero di transazioni gratuite limitate.

Per utilizzare nello specifico il servizio di opinion mining è stato necessario tradurre le recensioni in lingua inglese, poiché Azure mette a disposizione tale funzionalità solo per testi in lingua inglese. Di conseguenza, prima di dare in input la recensione al servizio, si è implementato uno script su Google Apps Script, per effettuarne la traduzione.

Per via di questo inconveniente alcuni degli aspetti rilevati potrebbero essere poco consistenti con la realtà, poiché l’operazione di traduzione non è efficiente al 100%.

* 1. **Rappresentazione grafica del workflow**



1. **Estrazione dei dati**

Dopo aver utilizzato Apify per ottenere i dati per le città selezionate, abbiamo creato una directory contentente questi file. Abbiamo creato un main Java scandire la lista dei file e per ciascuno di essi abbiamo letto le informazioni di nostro interesse e le abbiamo memorizzate in delle classi bean. Abbiamo utilizzato due tipi di bean: uno per i ristoranti e un altro per le recensioni, in particolare all’interno del bean per i ristoranti troviamo una lista di bean per le recensioni associate.

**public** **class** RistoBean {

**public** RistoBean(String nomeRistorante,ArrayList<String>cucina,**float** rate,ArrayList<ReviewsBean> recensioni, String city) {

**this**.nomeRistorante = nomeRistorante;

**this**.cucina = cucina;

**this**.rate = rate;

**this**.recensioni = recensioni;

**this**.city = city;

}

**public** **class** ReviewsBean {

**public** ReviewsBean(String titolo,String corpo,**float** rateReview,String userLocation) {

**this**.titolo = titolo;

**this**.corpo = corpo;

**this**.rateReview = rateReview;

**this**.userLocation = userLocation;

}

Il riempimento delle classi bean è stato svolto nel seguente modo:

**static** ArrayList<RistoBean> readJson(String name) **throws** JsonIOException, JsonSyntaxException, FileNotFoundException {

JsonParser parser = **new** JsonParser();

Object obj = parser.parse(**new** FileReader("file\_list\\" + name));

System.***out***.println(name);

JsonArray jsonArray = (JsonArray) obj;

ArrayList<RistoBean> arrayRisto = **new** ArrayList<RistoBean>();

**int** i = 0;

**for** (JsonElement element : jsonArray) {

**if**(i < 13) {

RistoBean risto = **new** RistoBean();

risto.setNomeRistorante(*getNomeRisto*(element));

risto.setRate(*getRatingRisto*(element));

risto.setCucina(*getCucina*(element));

ArrayList<String> cucina = risto.getCucina();

risto.setRecensioni(*getReviews*(element));

risto.setCity(name.substring(0,name.length()-5));

**if** (risto.getRecensioni().size()>0) {

arrayRisto.add(risto);

i++;

}

}

}

**return** arrayRisto;

}

1. **Sentiment analysis**

Una volta estratte le informazioni dai file, tutti i ristoranti sono stati memorizzati in un array di bean. Per ogni ristorante dell’array, abbiamo prelevato tutte le sue recensioni e per ciascuna di esse ne abbiamo fatto traduzione ed analisi del sentiment.

Per tradurre le recensioni in inglese, è stata usata una classe che, tramite un metodo, facesse richiesta ad uno script su Google Apps Script e restituisse in output la recensione tradotta dallo script.

**public** **class** Translator {

**public** Translator() {

}

**public** **static** String translate(String langFrom, String langTo, String text) **throws** IOException {

String urlStr =

"https://script.google.com/macros/s/AKfycbwqq5vTO3Ang0a4oitenKzA7MRg

VvHBRxuIzO5n2e2pKpG1FHcy/exec" +

"?q=" + URLEncoder.*encode*(text, "UTF-8") +

"&target=" + langTo +

"&source=" + langFrom;

URL url = **new** URL(urlStr);

StringBuilder response = **new** StringBuilder();

HttpURLConnection con = (HttpURLConnection) url.openConnection();

con.setRequestProperty("User-Agent", "Mozilla/5.0");

BufferedReader in = **new** BufferedReader(**new** InputStreamReader(con.getInputStream()));

String inputLine;

**while** ((inputLine = in.readLine()) != **null**) {

response.append(inputLine);

}

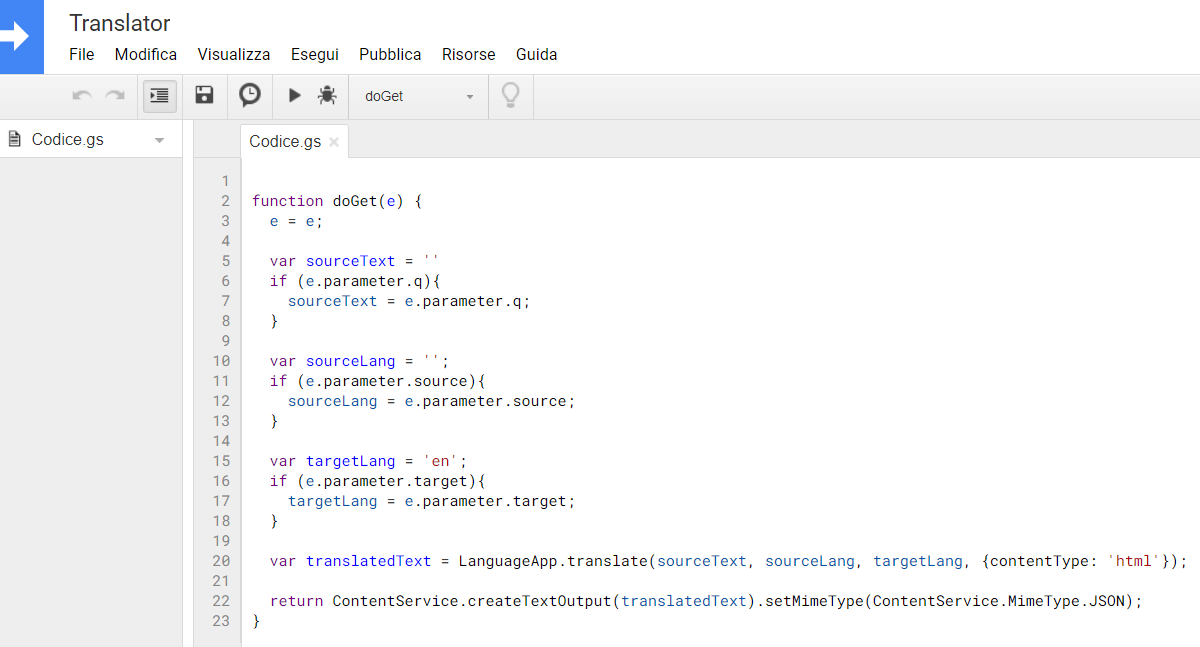
in.close();

**return** response.toString();

}

}

Il seguente script fa uso della classe LanguageApp, un servizio di Google per la traduzione di testo, che è capace di individuare automaticamente la lingua in input e tradurre il testo nella lingua target:



I risultati sono stati memorizzati in oggetti JSON per costituire un file di output.

JSONArray ristos = **new** JSONArray();

**for**(RistoBean r : array) {

JSONObject obj = **new** JSONObject();

obj.put("NOME",r.getNomeRistorante());

obj.put("CITTA'" , r.getCity());

obj.put("RANKING" ,r.getRate());

ArrayList<ReviewsBean> b = r.getRecensioni();

JSONArray recensioni = **new** JSONArray();

**for**(ReviewsBean t : b) {

JSONObject singleReview = **new** JSONObject();

String textOfReview = t.getTitolo()+" "+t.getCorpo();

**double** rateReview = t.getRateReview();

singleReview.put("RECENSIONE", textOfReview);

singleReview.put("RATE RECENSIONE", rateReview);

JSONObject sentimentInfo = **new** JSONObject();

sentimentInfo =

TextAnalyticsSamples.*sentimentAnalysisWithOpinionMining*(*client*,

textOfReview);

singleReview.put("SENTIMENT RECENSIONE", sentimentInfo);

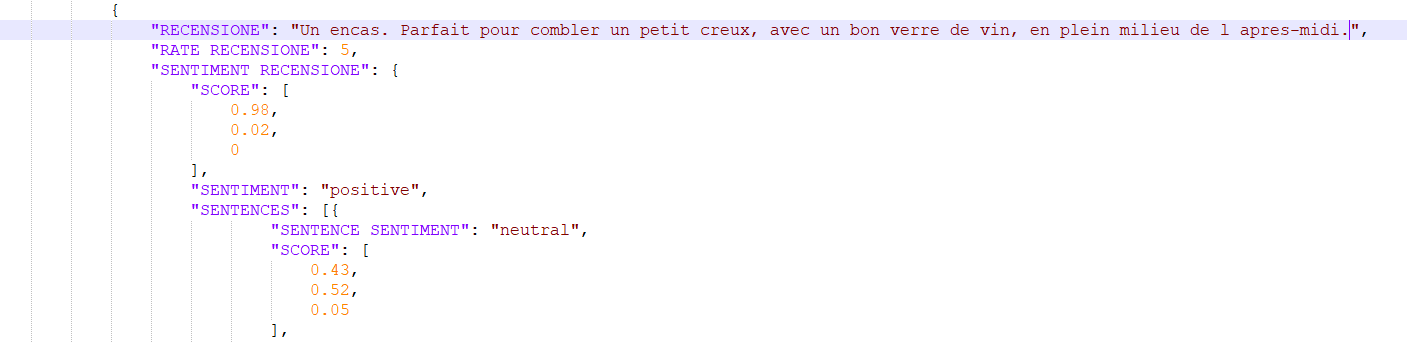
recensioni.put(singleReview);

}

obj.put("RECENSIONI", recensioni);

ristos.put(obj);

}

Una volta ottenuti i risultati dell’analisi, il JSONObject ottenuto è stato salvato all’interno di un JSONArray contenente tutti gli oggetti ristorante, così da poterli scrivere in un file JSON: 



* 1. **Azure Text Analysis and Opinion Mining**

Per analizzare la recensione, abbiamo prima prelevato il sentiment dell’intera recensione, ottenendo dei punteggi per quantificare l’aspetto sentimentale della recensione ed individuare se fosse una recensione positiva, negativa o neutrale.

AnalyzeSentimentOptions options = **new** AnalyzeSentimentOptions().setIncludeOpinionMining(**true**);

**final** DocumentSentiment documentSentiment = client.analyzeSentiment(doc, "en",

options);

SentimentConfidenceScores scores = documentSentiment.getConfidenceScores();

System.***out***.printf(

"\tRecognized document sentiment: %s, positive score: %f, neutral score: %f,

negative score: %f.%n",

documentSentiment.getSentiment(), scores.getPositive(), scores.getNeutral(),

scores.getNegative());

String sentimento = documentSentiment.getSentiment().toString();

**double** positiveSc = scores.getPositive();

**double** neutralSc = scores.getNeutral();

**double** negativeSc = scores.getNegative();

rootSentiment.put("SENTIMENT", sentimento);

sentInfo.put(positiveSc);

sentInfo.put(neutralSc);

sentInfo.put(negativeSc);

rootSentiment.put("SCORE", sentInfo);

Attraverso il metodo getSentences() di Azure, abbiamo diviso il commento in più parti, per onguna di queste sono stati calcolati i punteggi per individuare nuovamente gli aspetti sentimentali.

JSONArray sentences = **new** JSONArray();

documentSentiment.getSentences().forEach(sentenceSentiment -> {

SentimentConfidenceScores sentenceScores =

sentenceSentiment.getConfidenceScores();

System.***out***.printf("\t\tSentence sentiment: %s, positive score: %f,

neutral score: %f, negative score: %f.%n",

sentenceSentiment.getSentiment(), sentenceScores.getPositive(),

sentenceScores.getNeutral(), sentenceScores.getNegative());

JSONObject singleSentence = **new** JSONObject();

JSONArray sentenceSent = **new** JSONArray();

String sentenceInfo = sentenceSentiment.getSentiment().toString();

**double** sentencePos = sentenceScores.getPositive();

**double** sentenceNeut = sentenceScores.getNeutral();

**double** sentenceNeg = sentenceScores.getNegative();

singleSentence.put("SENTENCE SENTIMENT", sentenceInfo);

sentenceSent.put(sentencePos);

sentenceSent.put(sentenceNeut);

sentenceSent.put(sentenceNeg);

singleSentence.put("SCORE", sentenceSent);

Per ogni frase, tramite l’opinion mining di Azure, abbiamo colto gli aspetti, i quali permettono di comprendere il contesto della recensione e l’opinione del cliente in merito a quest’ultimo. In questo caso non si è calcolato un punteggio, ma si è estratta solamente l’emozione dominante.

JSONArray aspects = **new** JSONArray();

sentenceSentiment.getMinedOpinions().forEach(minedOpinions -> {

AspectSentiment aspectSentiment = minedOpinions.getAspect();

System.***out***.printf("\t\t\tAspect sentiment: %s, aspect text: %s%n",

aspectSentiment.getSentiment(), aspectSentiment.getText());

JSONObject singleAspect = **new** JSONObject();

/\*Raccolta ed inserimento degli aspetti, formato sentimento- testo\*/

String sentAspect = aspectSentiment.getSentiment().toString();

String textAspect = aspectSentiment.getText();

singleAspect.put("ASPECT", textAspect);

singleAspect.put("SENTIMENT ASPECT", sentAspect);

JSONArray opinions = **new** JSONArray();

JSONObject singleOpinion = **new** JSONObject();

**for** (OpinionSentiment opinionSentiment : minedOpinions.getOpinions())

{

singleOpinion = **new** JSONObject();

String minedSentiment =

opinionSentiment.getSentiment().toString();

String textMined = opinionSentiment.getText();

singleOpinion.put("OPINION", textMined);

singleOpinion.put("OPINION SENTIMENT", minedSentiment);

opinions.put(singleOpinion);

}

singleAspect.put("OPINION MINING", opinions);

aspects.put(singleAspect);

});

Tutti i dati estrapolati, sono stati inseriti all’interno di un JSONObject, che è stato poi restituito alla funzione principale per inserirlo nel JSONArray delle recensione per il ristorante considerato.

**6. Preprocessing dei risultati**

Una volta ottenuti tutti i file con i dati di analisi, sono stati scanditi al fine di suddividere i ristoranti in base alla città, considerando ogni città come appartenente ad una delle sequenti zone: nord, centro e sud Italia. Per ciascun ristorante, abbiamo contato sia la quantità di recensioni totali, sia la quantità di recensioni positive, negative e neutrali, in particolare abbiamo eseguito questa operazione sia nel contesto Nazionale che quello della singola zona.

Abbiamo inoltre individuato su quali aspetti ogni recensione si è espressa (cibo, location, servizi e costo) e contato il numero di valutazioni positive, negative e mixed per ognuno, anche qui sia da un punto di vista Nazionale che per suddivisione di zona.

**public** **static** **int**[] analyzeNord(String s,ArrayList<RistoBean> ristorantiNord) **throws** JsonIOException, JsonSyntaxException, FileNotFoundException {

JsonParser parser = **new** JsonParser();

Object obj = parser.parse(**new** FileReader("output\\" + s));

JsonObject risto = (JsonObject) obj;

JsonArray j = (JsonArray) risto.get("RISTORANTI");

**int** counterPositiveScoreNord = 0;

**int** counterNeutralScoreNord = 0;

**int** counterNegativeScoreNord = 0;

**int** counterMixedScoreNord = 0;

**for**(**int** i =0;i<j.size();i++) {

JsonObject x = (JsonObject) j.get(i);

String city = x.get("CITTA'").toString().replace("\"","");

String nameOfRest = x.get("NOME").toString().replace("\"","");

JsonArray recensioni = (JsonArray) x.get("RECENSIONI");

**if**(city.equals("bologna") || city.equals("genova") ||

city.equals("milano") || city.equals("torino") ||

city.equals("trieste") || city.equals("trento") ||

city.equals("venezia"))

{

**if**(!*isAlreadyInArray*(ristorantiNord,nameOfRest)) {

RistoBean beanResturant = **new** RistoBean();

beanResturant.setNomeRistorante(nameOfRest);

beanResturant.setCity(city);

ristorantiNord.add(beanResturant);

**for** (**int** k = 0;k<recensioni.size();k++) {

JsonObject oggettoRecensione = (JsonObject)

recensioni.get(k);

JsonObject temp = (JsonObject)

oggettoRecensione.get("SENTIMENT RECENSIONE");

String sentRev = temp.get("SENTIMENT").toString().replace("\"","");

*getAspects*(temp, *scoreFoodNord*, *scoreLocationNord*, *scoreServiceNord*, *scorePriceNord*);

**switch**(sentRev) {

**case** "positive":

counterPositiveScoreNord++;

**break**;

**case** "neutral":

counterNeutralScoreNord++;

**break**;

**case** "negative":

counterNegativeScoreNord++;

**break**;

**case** "mixed":

counterMixedScoreNord++;

**break**;

}

}

}

}

}

**int** [] score= **new** **int**[4];

score[0] = counterPositiveScoreNord;

score[1] = counterNeutralScoreNord;

score[2] = counterNegativeScoreNord;

score[3] = counterMixedScoreNord;

**return** score;

}

Per memorizzare le infomazioni così ottenute, abbiamo utilizzato dei bean Score, in particolare uno per ogni zona, tramite i quali abbiamo calcolato anche le percentuali e tramite i quali abbiamo calcolato i risultati Nazionali.

**public** **class** Score {

**private** **int** countPositive;

**private** **int** countNegative;

**private** **int** countNeutral;

**private** **int** countMixed;

**private** **int** totalCount;

**public** Score() {

**this**.countMixed = 0;

**this**.countNegative = 0;

**this**.countNeutral = 0;

**this**.countPositive = 0;

**this**.totalCount = 0;

}

**public** **void** addPositive() {

**this**.countPositive++;

**this**.totalCount++;

}

**public** **void** addNegative() {

**this**.countNegative++;

**this**.totalCount++;

}

**public** **void** addNeutral() {

**this**.countNeutral++;

**this**.totalCount++;

}

**public** **void** addMixed() {

**this**.countMixed++;

**this**.totalCount++;

}

**public** **float**[] getAvg() {

**float**[] res = **new** **float**[4];

**float** cps = (**float**) countPositive;

**float** cng = (**float**) countNegative;

**float** mxd = (**float**) countMixed;

**float** ttc = (**float**) totalCount;

res[0] = cps\*100/ttc;

res[1] = cng\*100/ttc;

res[2] = 0;

res[3] = mxd\*100/ttc;

**return** res;

}

}